Deployment Concerns in Machine Learning Systems:

Unintended Interactions and Accountability

by

Vasisht Duddu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2026

© Vasisht Duddu 2026

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:
Karthik Pattabiraman
Professor
Department of Electrical and Computer Engineering
University of British Columbia, Canada

Supervisor:
N. Asokan
University Professor
David R. Cheriton School of Computer Science
University of Waterloo, Canada

Internal Members:
Xi He
Associate Professor
David R. Cheriton School of Computer Science
University of Waterloo, Canada

Yaoliang Yu
Associate Professor
David R. Cheriton School of Computer Science
University of Waterloo, Canada

Internal-External Member:
Sirisha Rambhatla
Assistant Professor
Department of Management Science and Engineering
University of Waterloo, Canada

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The thesis is a substantially revised and extended version of various published papers. All projects were supervised by Prof. N. Asokan, and were proposed and led by Vasisht. All authors were involved in writing the papers. The remaining contributions of the co-authors across all the projects are described below.

Part 1: Unintended Interactions in Machine Learning (ML)

- **Chapter 3: *Unintended Interactions among ML Defenses and Unrelated Risks***
  This chapter is based on *"SoK: Unintended Interactions among Machine Learning Defenses and Risks"* (published at IEEE S&P 2024) with Sebastian Szyller from Aalto University.
  - Vasisht collected and systematized all the papers.
  - Vasisht and Sebastian explored the relation of unintended interactions with the factors.
  - Vasisht designed and executed the experiments, and analyzed the results.
  - Sebastian provided expert advice.
  - Lipeng He (non-author) executed the new experiment (not in the published paper).

  V. Duddu, S. Szyller, N. Asokan. **SoK: Unintended Interactions among Machine Learning Defenses and Risks.** *IEEE Symposium on Security and Privacy (S&P)*, 2024 [1].

- **Chapter 4: *Combining ML Defenses without Conflicts***
  This chapter is based on the paper *"Combining Machine Learning Defenses without Conflicts"* (published at TMLR 2025) with Rui Zhang from Zhejiang University, who was a visiting Ph.D. student at the University of Waterloo.
  - Rui implemented and ran the experiments for three defenses.
  - Vasisht implemented and ran the experiments for the remaining eight defenses.

  V. Duddu, R. Zhang, N. Asokan. **Combining Machine Learning Defenses without Conflicts.** *Transactions on Machine Learning Research (TMLR)*, 2025 [2].

- **Chapter 5: *Colluding Adversaries in ML Pipelines***
  This chapter is based on the paper *"SoK: Colluding Adversaries in ML Pipelines"* (published at USENIX SEC 2026) with Lipeng He and Asim Waheed from University of Waterloo.
  - Vasisht proposed and developed the framework, and systematized prior work.
  - Vasisht designed the experiments; Asim and Lipeng implemented and executed them.

  V. Duddu, L. He, A. Waheed, N. Asokan. **SoK: Colluding Adversaries in Machine Learning Pipelines.** *USENIX Security Symposium (USENIX SEC)*, 2026 [3].

Part 2: Accountability in Machine Learning Pipelines

- **Chapter 7: *Software-based ML Property Attestation Mechanisms***
  This chapter is based on the paper *"Attesting Distributional Properties of Training Data for Machine Learning"* (published at ESORICS 2024) with Anudeep Das from University of Waterloo, and Nora Khayata, Hossein Yalame, and Thomas Schneider from TU Darmstadt. This work was jointly supervised by Thomas Schneider.
  - Vasisht implemented the initial prototype for ML-based attestation mechanism.
  - Anudeep implemented and ran the evaluation pipeline, and the robustness experiments.
  - Vasisht developed the theory for the hybrid approach with help from Anudeep.
  - Nora and Hossein implemented and evaluated cryptographic property attestation.

  V. Duddu, A. Das, N. Khayata, H. Yalame, T. Schneider, N. Asokan. **Attesting Distributional Properties of Training Data for Machine Learning.** *European Symposium on Research in Computer Security (ESORICS)*, 2024 [4].

- **Chapter 8: *Hardware-based ML Property Attestation Mechanisms***
  This chapter is based on the paper *"`Laminator`: Verifiable ML Property Cards using Hardware-assisted Attestations"* (published at ACM CODASPY 2025) with Oskari Järvinen and Lachlan Gunn from Aalto University. Prach Chantasantitam and Asim Waheed (non-authors) helped with setting up the server, and executing the experiments.
  - Vasisht implemented the code to train and evaluate ML models.
  - Asim, Oskari, and Lachlan set up the SGX server.
  - Oskari and Lachlan implemented the attestations.
  - Vasisht integrated the attestations into the ML pipeline.
  - Prach ran the experiments and collected the results from an SGX server.
  - Vasisht and Lachlan parsed and analyzed the results.
  - All authors were involved in designing the experiments.

  V. Duddu, O. Järvinen, L. Gunn, N. Asokan. **`Laminator`: Verifiable ML Property Cards using Hardware-assisted Attestations.** *ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2025 [5].

Note: I acknowledge the use of a large language model (ChatGPT) exclusively for grammar and spelling correction purposes. All the text was written by me and refined for grammar and spelling. None of the text in the thesis was generated using the model.

Abstract

Machine learning (ML) models are increasingly being deployed for client-facing services (e.g., chatbots, search engines, and browsers), high-stakes decision-making applications (e.g., healthcare and criminal justice), and as part of larger systems (e.g., autonomous vehicles and operating systems). However, to deploy ML models for a particular application, practitioners need to address various deployment concerns including (i) *infrastructure issues* (e.g., latency, throughput, interoperability, scalability), (ii) *model design* (e.g., high utility and generalization, minimal overfitting, hyperparameter tuning, data processing), (iii) *environmental impact* (e.g., reducing carbon emissions, water and power consumption by data centers), (iv) *adversarial and societal risks* (e.g., security, privacy, safety, unfairness, poor transparency, misalignment, misinformation, and cyberattacks), and (v) *enabling governance* (e.g., verifying claims by practitioners, and regulatory compliance). I focus on two deployment concerns: *adversarial and societal risks*, and *enabling governance*, and address *unintended interactions* and *accountability* within these respective concerns. I present them as two parts of the thesis.

(Part 1) Unintended Interactions in ML: Substantial prior work explores the design of defenses against individual risks to security, privacy, fairness, transparency, and safety. I argue that this is not sufficient for real-world ML models that must protect against multiple risks simultaneously. Practitioners need to address additional challenges that emerge when doing so, including *unintended interactions*. A systematic understanding of such interactions is lacking, and I study three unintended interactions: (a) a defense against one risk may increase or decrease other unrelated risks; (b) conflicts among defenses can decrease their effectiveness when combined; and (c) potential for collusion among adversaries can enable executing an attack to amplify others. I propose frameworks to identify factors underlying such interactions, and present guidelines to conjecture about unexplored ones.

(Part 2) Accountability in ML Pipelines: Practitioners' claims about executing various ML operations needs verification by a verifier (e.g., regulator). This includes demonstrating ML properties covering the model, its training process, its training data, as well as deploying defenses and accounting for unintended interactions from **Part-1**. Such claims are currently communicated via *ML property cards* (e.g., model, data, and inference cards). I propose *ML property attestation mechanisms* that allow provers (e.g., model trainers) to demonstrate ML properties to verifiers, while ensuring model and data confidentiality. I show that existing *software-based mechanisms* are either inefficient (e.g., cryptographic mechanism), or ineffective and easily evaded (e.g., ML-based mechanism). I then identify *hardware-based mechanisms* using *trusted execution environments* as an efficient and effective alternative for providing ML property attestations. These attestations can then be used for *verifiable ML property cards*, to ensure accountability for practitioners' claims.

Acknowledgements

As I complete my Ph.D., I look back with gratitude on the past six years of my graduate studies. I am deeply grateful for the people I have met and learned from during this time, each of whom has played a role in bringing this thesis to completion.

To my advisor, Prof. N. Asokan, thank you for giving me the opportunity to join your group six years ago as a master's student. Beyond learning how to conduct rigorous research, I learned the importance of humility and integrity in navigating professional challenges. Your patience and empathy during difficult times helped me persevere, and I am deeply grateful for your training, support, and mentorship.

To my committee members (Karthik Pattabiraman, Xi He, Yaoliang Yu, Sirisha Rambhatla), thank you for your time and insightful feedback, which greatly improved my thesis. *To my mentors* (Thomas Schneider, Xi He, Antoine Boutet, Yan Shvartzshnaider, Ghassan Karame, Debasis Samanta, and Valentina Balas), I am grateful for your guidance and support throughout my career. Your research and writing styles have influenced mine over the years.

To my labmates, my time in the lab was made especially enjoyable by our stimulating discussions, memorable coffee runs, and shared meals. I will cherish these moments and friendships for years to come. *To all my friends*, my graduate studies over the past six years was enriched by the wonderful memories we shared. I would have been lost without all of you. Our trips, movies, parties, board-game nights, and potlucks, made my time unforgettable and gave me much-needed breaks from work.

To my family, I am deeply grateful for your unwavering support throughout my graduate studies, which allowed me to focus fully on my work. Thank you, Amma, for showing me the value of humility, kindness, and patience. Thank you, Nanna, for instilling in me a strong work ethic and fostering my interest in research, including through the many dinner-table conversations that ultimately led me to graduate school. Thank you, Siddharth, for lightening the mood and being a strong support to our parents by being close to home.

Finally, I have been fortunate to receive support from several fellowships and grants, including the ML and Systems Rising Star, IBM Ph.D. Fellowship, David R. Cheriton Scholarship, and Mastercard's Cybersecurity and Privacy Graduate Excellence Award. I am deeply grateful for this support and for the confidence placed in me. As I embark on the next chapter of my career, I am reminded of the words often (mis)attributed to A. A. Milne (in *Winnie-the-Pooh*): "*How lucky I am to have something that makes saying goodbye so hard.*"

Dedication

To my parents, whose sacrifices, and unwavering love
and support, made everything possible.

Table of Contents

List of Figures xiv

List of Tables xvi

List of Abbreviations xxii

List of Symbols xxv

1 Introduction 1

1.1 **Part-1**: Unintended Interactions in Machine Learning 2

1.2 **Part-2**: Accountability in Machine Learning Pipelines 5

Part 1: Unintended Interactions in Machine Learning

2 Background 9

2.1 Machine Learning Pipeline and Notations 9

2.2 Overfitting and Memorization in Machine Learning 12

2.3 Streamlined Threat Model for Machine Learning 15

2.4 Defenses for Machine Learning Models 21

3 Unintended Interactions among ML Defenses and Unrelated Risks **26**
3.1 Framework to Explore Interactions . . . 27
3.2 Understanding Unintended Interactions . . . 30
3.2.1 Revisiting Defenses for ML . . . 31
3.2.2 Revisiting Risks in ML . . . 33
3.2.3 Surveying Unintended Interactions . . . 37
3.2.4 Guideline: Exploring Unintended Interactions . . . 46
3.3 Unexplored Interactions and Conjectures . . . 47
3.3.1 `GpFair` (Group Fairness) → `DtRecon` (Data Reconstruction) . . . 48
3.3.2 `GpFair` (Group fairness) → `DistInf` (Distribution Inference) . . . 51
3.3.3 `Expl` (Explanations) → `DistInf` (Distribution Inference) . . . 53
3.3.4 Other Unintended Interactions . . . 55
3.4 Discussions . . . 56

4 Combining ML Defenses without Conflicts **58**
4.1 Systematization of Related Work . . . 60
4.2 Insights from Systematization . . . 66
4.3 Design of `Def\Con` (Proposed Combination Technique) . . . 68
4.4 Experimental Setup . . . 70
4.4.1 Datasets and Models . . . 70
4.4.2 Choosing Defenses for Evaluation . . . 70
4.4.3 Threat Models for Defenses . . . 71
4.4.4 Metrics and Implementations . . . 73
4.5 Evaluation . . . 76
4.5.1 Effectiveness of Individual Defenses . . . 76
4.5.2 Accuracy against Prior Work . . . 76
4.5.3 Accuracy via Empirical Evaluation . . . 79
4.5.4 Hyperparameter Tuning for Combinations . . . 81
4.5.5 Other Requirements . . . 84
4.6 Discussions . . . 86

5 Colluding Adversaries in ML Pipelines **88**
5.1 Framework to Explore Colluding Adversaries . . . 90
5.1.1 Factors Enabling Collusion . . . 90
5.1.2 Relation between Attacks and Factors . . . 93
5.1.3 Guideline to Predict Collusion . . . 97
5.2 Systematization of Prior Work . . . 98
5.2.1 Train-Inference Collusion . . . 99
5.2.2 Inference-time Collusion . . . 100
5.3 Unexplored Collusion among Adversaries . . . 101
5.3.1 Conjectures for Unexplored Train-Inference Collusion . . . 101
5.3.2 Conjectures for Unexplored Inference-time Collusion . . . 103
5.4 Accounting for Adversary's Characteristics . . . 108
5.5 Discussions . . . 110

Part 2: Accountability in Machine Learning Pipelines

6 Background **115**
6.1 Datasheets and Model Cards . . . 115
6.2 Cryptographic Primitives . . . 116
6.3 Trusted Execution Environments . . . 118

7 Software-based ML Property Attestation Mechanisms **120**
7.1 ML Property Attestation . . . 122
7.2 Case Study: Distributional Equity . . . 124
7.3 Experimental Setup . . . 128
7.4 Experimental Evaluation . . . 131
7.4.1 Mechanism 1: ML-based . . . 131
7.4.2 Mechanism 2: Cryptographic . . . 134
7.4.3 Mechanism 3: Hybrid . . . 135
7.5 Discussions . . . 139

8 Hardware-based ML Property Attestation Mechanisms **140**
8.1 Hardware-based Attestations for Property Cards . . . 141
8.2 Laminator: Framework Design . . . 146
8.3 Experimental Setup . . . 150
8.4 Evaluation . . . 153
8.4.1 **R1**: Effective . . . 153
8.4.2 **R2**: Efficiency . . . 153
8.4.3 **R3**: Scalability . . . 157
8.4.4 **R4**: Versatility . . . 157
8.4.5 **R5**: Robustness . . . 158
8.5 Discussions . . . 158

9 Summary and Future Directions **160**
9.1 Summary of Contributions . . . 160
9.2 Lessons Learned . . . 163
9.3 Future Research Directions . . . 165
9.3.1 Chapter-wise Future Directions . . . 165
9.3.2 Broader Future Directions . . . 169
9.4 Concluding Remarks . . . 172

References **175**

Appendices **233**
A Chapter 4: Combining ML Defenses without Conflicts . . . 233
A.1 Formal Analysis of `Def\Con` . . . 233
A.2 Formal Analysis for Multi-way Combination Algorithm . . . 235
B Chapter 7: Software-based ML Property Attestation . . . 235
B.1 Selecting Window Size . . . 235

B.2 Effectiveness of Robust f_{att} . . . 237
B.3 Impact of Adversarial Fine-Tuning Attack on $\mathcal{M}$'s Accuracy . . . 238
B.4 Details for Cryptographic Mechanism . . . 238
B.5 Hybrid Attestation: FAR vs. Cost on $\mathcal{D}_{\mathrm{P}}^{vrf}$ . . . 240

List of Figures

2.1 **Overview of ML pipeline**: Raw data from data owners (DtOwnr) is aggregated by a data provider (DtProv) and processed into training ($\mathcal{D}_{tr}$), validation ($\mathcal{D}_{val}$), and test ($\mathcal{D}_{te}$) datasets. A model trainer (ModTrnr) then trains or fine-tunes a model $\mathcal{M}$ using an architecture or pre-trained model from a model provider (ModProv), code for the training algorithm from a code provider (CodeProv), and a training configuration specified by ModTrnr. The trained $\mathcal{M}$, owned by a model owner (ModOwnr), is evaluated on $\mathcal{D}_{te}$ and subsequently deployed by a service provider (SrvProv) for clients (Clnt) to send queries and get responses. 10

2.2 **Relationship between overfitting and memorization:** Average mem across all data records in $\mathcal{D}_{tr}$ and overfitting ($\mathcal{G}_{err}$) are at the bottom. Circles indicate data records in $\mathcal{D}_{tr}$, and crosses indicate data records in $\mathcal{D}_{te}$. 14

3.1 **Distinguishability of model observables across demographic groups** (O2.2) decreases when fairness constraints are applied $\mathcal{M}^{gpfr}$. 50

3.2 **Nature of interaction (Expl $\rightarrow$ DistInf):** Attack accuracy of DistInf to differentiate explanations from model trained on $\mathcal{D}_{tr}$ with α_1=0.5 and $\alpha_2 \in \{0.1 - 0.9\}$. 54

4.1 **Flowchart depicting various steps in Def\Con** to identify a conflict between $\texttt{Def}_1$ and $\texttt{Def}_2$ ($\texttt{Def}_2$ is applied after $\texttt{Def}_1$). 68

6.1 **Overview of ML property cards:** Datasheets describe properties of $\mathcal{D}_{tr}$ (e.g., bias and attribute distribution), model cards document training and evaluation details (e.g., how and on what the model was trained), and inference cards capture inference-time behavior (e.g., input–model–output relationships and applied filters). 116

7.1 **Inference-based attestation**: During preparation, $\mathcal{V}er$ trains f_{att} using the first layer parameters of models trained on $\mathcal{D}_{tr}^{prv}$ with p_{req} ($\{\mathcal{M}_{p_{req}}^{i}\}_{i=1}^{\mathcal{N}}$) and $!p_{req}$ ($\{\mathcal{M}_{!p_{req}}^{i}\}_{i=1}^{\mathcal{N}}$). During attestation, $\mathcal{V}er$ uses the first layer parameters of $\mathcal{M}$ to attest whether it was indeed trained on $\mathcal{D}_{tr}^{prv}$ with p_{req} or not. 125

7.2 **Cryptographic attestation**: $\mathcal{P}rv$ sends the secret shares of $\mathcal{D}^{\mathcal{P}rv}$ to servers $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$. The servers securely compute `DistCheck` for $\mathcal{D}^{\mathcal{P}rv}$ and train $\mathcal{M}_{2pc}$ on it with their secret shares using 2PC. The output shares are sent to $\mathcal{V}er$ for reconstruction. 127

7.3 **Fixed FAR@5%: Expected cost on $\mathcal{D}_{\mathbf{P}}^{vrf}$**: ω_{crpt} is a placeholder for ω_{crpt}^{comp} and ω_{crpt}^{comm}. Expected cost is less than cryptographic attestation (=ω_{crpt}). 135

7.4 **Fixed FRR@5%: Trade-off between FAR and expected cost on varying $\mathcal{N}_{spchk}$ on $\mathcal{D}_{te}^{\mathcal{V}er}$**: Expected cost is less than cryptographic mechanism (=ω_{crpt}) and effectiveness is better than ML-based mechanism. ω_{crpt} is a placeholder for both ω_{crpt}^{comp} and ω_{crpt}^{comm}. 137

8.1 **Overview of `Laminator`'s design:** Components already existing or part of the infrastructure are indicated in gray, while components specific to `Laminator` are indicated in yellow. There can be separate enclaves for each attestation type, which are generated by changing the "measurer" (in green). Dashed lines correspond to assertions by a trusted certifier, while solid lines correspond to assertions by a TEE. 147

8.2 **Overview of ML property attestations:** The inputs and outputs for various attestations are described along with the content of the attestations. 149

9.1 In addition to identifying and mitigating risks within individual pillars of trust, trustworthy ML deployment requires a holistic systems analysis (e.g., unintended interactions), and supporting governance, (e.g., for accountability and regulatory compliance). 173

B.2 **Fixed FRR@5%: Expected cost on varying $\mathcal{N}_{spchk}$ on $\mathcal{D}_{\mathbf{P}}^{vrf}$.** $\mathcal{V}er$ can use suitable value of $\mathcal{N}_{spchk}$ identified from $\mathcal{D}_{te}^{\mathcal{V}er}$ to note the corresponding FAR and expected cost. 241

List of Tables

1.1 **Overview of chapters and their titles, corresponding research questions ("RQs"), references ("Ref."), and their statuses.** 7

2.1 **Summary of the characteristics of $\mathcal{A}dv$ and their descriptions**, from which the elements of the sets constituting `TM` are drawn. 17

2.2 **Summary of risks and defenses, and their impact on accuracy:** I indicate defenses at different stages of the ML pipeline (pre-, in-, post-training). Column "ϕ_{acc}" indicates the impact of the defense on ϕ_{acc} where "$\vee$" $\rightarrow$ decrease, "$\sim$" $\rightarrow$ no effect, and "$\wedge$" $\rightarrow$ increase. 24

3.1 **Summary of factors in the framework:** I include factors that influence overfitting and memorization, covering characteristics of data, model, and objective function. 30

3.2 **Relation between factors with defenses/risks:** Correlation between the effectiveness of a defense `Def` and susceptibility to a risk `Rk` with a factor `<f>`: $\uparrow$ indicates `Def` or `Rk` positively correlates with `<f>`; $\downarrow$ indicates a negative correlation. 34

3.3 **Unintended interactions with security defenses**: *Type of interaction (I)*: $\bullet\!\!\rightarrow$ risk increases; $\bullet\!\!\rightarrow$ risk decreases; $\bullet\!\!\rightarrow$ unexplored. I indicate exceptions to these under "References" and discuss them in the text. *Underlying factors:* $\bullet \rightarrow$ empirical; $\circledast \rightarrow$ theoretical and $\bigcirc \rightarrow$ conjectured. Under `02`, I write `02.1`, `02.2`, and `02.3` as 1, 2, and 3 respectively. I mark the factors evaluated for conjectured interactions as $\ast$ (See Section 3.3). 38

3.4 **Unintended interactions with defenses for privacy, fairness, and transparency**: Symbol and their descriptions are the same as in Table 3.3. 42

3.5 **Summary of unexplored interactions:** These interactions are marked as ● in Table 3.3 and 3.4, and the interaction between a defense to a risk with "⟶" between them. 48

3.6 **Nature of interaction:** Comparison of ϕ_{acc} on $\mathcal{D}_{te}$, fairness (AUC scores, p%-rule values) and reconstruction loss for $\mathcal{M}$ and $\mathcal{M}^{gpfr}$ across ten runs. ⇑ indicates higher value is desirable and ⇓ indicates lower value is desirable. 50

3.7 **Number of input attributes (D3):** Reconstruction loss and ϕ_{acc} on $\mathcal{D}_{te}$ for different number of input attributes for $\mathcal{M}$ and $\mathcal{M}^{gpfr}$ (averaged across ten runs). 51

3.8 **Nature of interaction (Discr → DistInf)**: I report the attack accuracy where $\alpha_1 = 0.5$ and vary α_2. I compare the attack accuracy on $\mathcal{M}$ (as baseline) to $\mathcal{M}^{gpfr}$, where a value close to 50% indicates random guess accuracy. 52

3.9 **Number of input attributes (D3):** Attack accuracy for $\alpha_1 = 0.5$ and $\alpha_2 = 0.1$ while varying the number of input attributes (results averaged across ten runs). 54

3.10 **Model capacity (M1):** Attack accuracy on explanations corresponding to α_1=0.5 and α_2=0.1 for different models (averaged across ten runs). 55

4.1 **Overview of pairwise combinations among defenses**: For each combination cell, I cite related work and indicate the "interaction type" (● → alignment, ● → conflict, or ● → unexplored), and "combination technique" used (**T1** or **T2**). 62

4.2 **Requirements satisfied by the combination techniques**: ○ → Not satisfied; ⊛ → Partially satisfied; ● → Fully satisfied. 67

4.3 **Effectiveness of defenses:** For metrics, I use ↑ (resp. ↓) where a higher (or lower) value is better, and "x" is shorthand for $\phi_{(x)}$. (ϕ_{acc} is for context). 77

4.4 **Decision points for Def\Con:** For each defense, I identify key parameters used in evaluating combination effectiveness (Figure 4.1): type of change in **S2** (Global, Local, or None); whether the defense uses a risk in **S3** ("Yes" for backdoors or adversarial examples); and whether the defense protects against a risk in **S4**. 78

4.5 **Evaluating combinations [C9-C24]:** For brevity, "x" in **Metric** column is shorthand for $\phi_{(x)}^{\hat{\texttt{Def}}}$ where $\hat{\texttt{Def}}$ is the combination. I use ↑ (resp. ↓) to indicate if a higher (lower) value is better. For defense effectiveness, I use green when $\phi_{(.)}^{\hat{\texttt{Def}}}$ is better or equal to "single defense" baseline; orange for better or equal to "single defense" but worse than "no defense"; red for worse than "no defense". For technique predictions, I use symbol Δ (resp. Ψ) to refer to Def\Con (naïve technique) and a color code green (resp. red) to indicate alignment (conflict) among defenses. $\texttt{Def}_1$ and $\texttt{Def}_2$ indicate order of applying defenses. (ϕ_{acc} is for context). 79

4.6 **Evaluating combinations [C25-C38]:** The descriptions of the symbols and colors are the same as in Table 4.5. 80

4.7 **Hyperparameter configurations** for tuning defenses in conflicting combinations. 82

4.8 **Scalability (R2) of Def\Con to $\texttt{Def}_1$, $\texttt{Def}_2$, and $\texttt{Def}_3$ (in order):** Color coding and notations are the same as in Tables 4.5 and 4.6. 85

5.1 **Factors enabling collusion and their descriptions** 93

5.2 **Relation of attacks and factors for train-inference collusion:** I use ↑ (or ↓) to indicate positive (negative) correlation. Rows for Poisn under "How effectiveness of attack (Poisn) leads to change in factors" (left half) indicates how Poisn influences a change in factors, instead of how change in factors influence effectiveness of Poisn (as in Chapter 3). Rows under "How changes in factors correlate with attack effectiveness" (right half) are reproduced from Chapter 3; only the factors that can be manipulated by $\mathcal{A}dv$ are considered. Gray → not applicable; empty cells → no prior literature reported the correlations. 94

5.3 **Relation of attacks and factors for inference-time collusion:** I use ↑ (or ↓) to indicate positive (negative) correlation. I exclude Discr, as it does not correspond to any attack that $\mathcal{A}dv$ can exploit. Note that the factors here are related to the $\mathcal{A}dv$'s *knowledge* inferred from the first attack (as outcomes) that satisfies the prerequisite for the second. 95

5.4 **Train-inference collusions:** Collusion between $\mathcal{A}dv1$ (where $\texttt{Att}_1$ is Poisn) and $\mathcal{A}dv2$ (corresponding to $\texttt{Att}_2$). **"Factors"** → common enabling factors (from Table 5.2) after applying the guideline to check for potential for collusion (↑, ↑). Gray → no prior work. 99

5.5 **Inference-time collusions:** Both adversaries perform inference-time attacks: $\texttt{Att}_1$ and $\texttt{Att}_2$. **"Factors"** → common enabling factors (from Table 5.3) identified after applying the guideline to check for collusion potential (prerequisite of $\texttt{Att}_2$ is satisfied by $\texttt{Att}_1$ indicated as ↑, ↑). Gray → no prior work. 100

5.6 `Poisn` → `ModExt`: Surrogate model's accuracy ("Surr. Acc.") and fidelity ("Surr. Fid.") for 0% poison are the baseline. For a given budget, green → higher than baseline, orange → similar to baseline (within std. dev.), and red → lower than baseline. I report target accuracy to show that `ModExt` is effective for different poisoning rates. 102

5.7 `ModExt` → `DistInf`: *Baseline* → shadow models trained independently without querying $\mathcal{M}$ (target); *Cross-Arch* → VGG11 shadow model extracted from $\mathcal{M}$; *Same-Arch* → ResNet34 shadow model extracted from $\mathcal{M}$. Green → higher than baseline, orange → similar to baseline (within std. dev.), red → lower than baseline. 105

5.8 **Impact of $\mathcal{A}dv$'s characteristics on collusion potential ($\mathcal{A}dv$1 → Change; $\mathcal{A}dv$2 → Same):** Rows correspond to $\texttt{Att}_1$; we vary $\mathcal{A}dv$1's characteristics while $\mathcal{A}dv$2 is fixed; values range from weak (left) to strong (right). ★ → minimum requirement for $\mathcal{A}dv$1 to execute $\texttt{Att}_1$; ➤ → collusion potential with $\texttt{Att}_2$ increases by strengthening the characteristic from ★; empty cell → insufficient, unrealistic, or unnecessary. Refer to Chapter 2 for description of notations. Markings are based on prior work covering each of the attacks [6, 7, 8, 9, 10, 11]. 108

5.9 **Impact of $\mathcal{A}dv$'s characteristics on collusion potential ($\texttt{Att}_2$):** Rows correspond to $\texttt{Att}_2$; I vary $\mathcal{A}dv$2's characteristics while $\mathcal{A}dv$1 is fixed. Strengthening $\mathcal{A}dv$2 raises effectiveness directly (not via collusion, hence omitted); I instead mark its effect on collusion potential with the listed $\texttt{Att}_1$ (s): ↓ → collusion potential decreases with characteristic; ✗ → collusion not required as characteristic meets the prerequisite factor. Columns under $\overline{\texttt{k}_\texttt{m}}$ are left empty for `ModExt`, as they are the outcomes of `ModExt`. 109

7.1 **Summary of datasets:** I describe the classification task, the attested property, whether it is explicit in $\mathcal{D}_{tr}$, and the dataset sizes for $\mathcal{V}er$ and $\mathcal{P}rv$ before sampling. 129

7.2 **TAR and TRR with 5% thresholds for FAR and FRR, respectively, along with EER across different p_{req} windows on $\mathcal{D}_{\mathbf{P}}^{vrf}$:** The p_{req} value within the window is indicated in **bold**. Edge p_{req} values have higher effectiveness than middle p_{req} values due to higher distinguishability in first layer parameters [12]. 132

7.3 **Robustness against first layer parameter perturbations with and without a defense:** Utility (ϕ_{acc}) is calculated using AUC on FAR-TAR for a clean dataset. FAR $\rightarrow$ lack of robustness. Green $\rightarrow \phi_{\text{acc}}$ decreases or FAR $< 5\%$, red $\rightarrow \phi_{\text{acc}}$ increases or FAR $\geq 5\%$. 133

7.4 **Computation (ω_{crpt}^{comp}) and communication costs (ω_{crpt}^{comm}) of cryptographic attestation** for a single $\mathcal{P}rv$ averaged over 20 runs. 134

7.5 **Summary of requirements satisfied by various software-based ML property attestation mechanisms**: ○ $\rightarrow$ Not satisfied; ⊛ $\rightarrow$ Partially satisfied; ● $\rightarrow$ fully satisfied. 138

8.1 **Summary of models across different datasets** 152

8.2 **Performance of distributional property attestation:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). I omit "Output Size" as the output (i.e., distribution value) is a single value of negligible size. 154

8.3 **Performance of proof of training:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). 155

8.4 **Performance of accuracy attestation:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). I omit "Output Size" as the output (i.e., accuracy) is a single value of negligible size. 156

8.5 **Performance of proof of inference:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). I omit "Output Size" as the output (i.e., prediction) is a single value of negligible size. I report the overhead computed using attestation ("w/ att") and that with signing using a keypair ("w/ sgn"). 156

B.1 **Identifying the best window size (w)** using AUC scores to maximize attestation success. red → AUC score <0.85; green → AUC score >=0.85. $\pm 1 \rightarrow$ the window $\{p_{req}\text{-}1, p_{req}, p_{req}\text{+}1\}$ but only two of those ratios are combined for the edge values. Performance increases when using a window size of ± 1 instead of 0. ARXIV and BONEAGE do well across different p_{req} values, but CENSUS-S and CENSUS-R have low effectiveness for certain p_{req} values. 236

B.2 **Robust f_{att} effectiveness using TAR, TRR with 5% thresholds for FAR, FRR along with EER across different p_{req} windows on $\mathcal{D}_{\mathbf{P}}^{vrf}$:** The p_{req} value within the window is indicated in **bold**. We observe a similar trend as before: edge p_{req} values have higher effectiveness than middle p_{req} values. . . . 237

B.3 **Average accuracy across 1000** shadow models w/ and w/o malicious $\mathcal{P}rv$ modifying the first layer parameters followed by fine-tuning. We refer to them as $\mathcal{A}_{orig}$ and $\mathcal{A}_{att}$, respectively. green indicates $\mathcal{A}_{orig} >> \mathcal{A}_{att}$, orange indicates $\mathcal{A}_{orig} > \mathcal{A}_{att}$ (<2%) and red indicates $\mathcal{A}_{orig} \sim \mathcal{A}_{att}$. Across most p_{req} values, malicious $\mathcal{P}rv$ can restore $\mathcal{A}_{att}$ close to $\mathcal{A}_{orig}$. 239

List of Abbreviations

`1Shot` Adversary has one-shot to perform attack on target model
`Adpt` Adversary uses feedback from target model for attacks
`AttInf` Attribute inference risk
`BBox` Blackbox access to target model
`Bkdr` Backdoor risk
`Blind` Auxiliary data does not include labels
`Clnt` Client
`CodeProv` Code provider
`DiffPriv-In` In-training defense for differential privacy
`DiffPriv-Post` Post-training defense for differential privacy
`DiffPriv-Pre` Pre-training defense for differential privacy
`DiffPriv` Differential privacy
`Discr` Discrimination risk
`DistInf` Distribution inference risk
`DtOwnr` Data owner
`DtProv` Data provider
`DtRecon` Data reconstruction risk
`DtWM-Pre` Pre-training data watermarking
`DtWM` Data watermarking against unauthorized data usage
`EvsnRob-In` In-training defense against evasion
`EvsnRob-Post` Post-training defense against evasion
`EvsnRob-Pre` Pre-training defense against evasion
`EvsnRob` Defense against evasion
`Evsn` Evasion risk
`Expl-Post` Post-training defense for model explanations
`Expl` Defense for model explanations
`Fngrprnt-Post` Post-training defense for fingerprinting
`Fngrprnt` Defense for fingerprinting

`FullOvlap` Full overlap of auxiliary data with training data
`GBox` Graybox access to target model
`GpFair-In` In-training defense for group fairness
`GpFair-Post` Post-training defense for group fairness
`GpFair-Pre` Pre-training defense for group fairness
`GpFair` Defense for group fairness
`HbC` Honest-but-curious adversary who uses observable data for attacks without modifications
`KShot` Adversary can send k-queries to target model
`Mal` Malicious adversary who actively deviates from expected computations
`MdlWM-In` In-training model watermarking
`MdlWM-Post` Post-training model watermarking
`MdlWM-Pre` Pre-training model watermarking
`MdlWM` Model watermarking against unauthorized model ownership
`MemInf` Membership inference risk
`ModExt` Model extraction risk
`ModOwnr` Model owner
`ModProv` Model provider
`ModTrnr` Model trainer
`NoAccess` No Access
`NoBlind` Auxiliary data includes labels
`NoOvlap` No overlap between auxiliary data with training data
`PartOvlap` Partial overlap of auxiliary data with training data
`PoisnRob-In` In-training defense against poisoning/backdoors
`PoisnRob-Post` Post-training defense against poisoning/backdoors
`PoisnRob-Pre` Pre-training defense against poisoning/backdoors
`PoisnRob` Defense against poisoning/backdoors
`Poisn` Poisoning risk
`SrvProv` Service provider
`UnauthData` Unauthorized data usage
`Unlmtd` Adversary can send unlimited queries to target model
`WBox` Whitebox access to target model
`fullPred` Adversary has access to model's full prediction vector
`hardLbl` Adversary has access to hard labels from models
`nonAdpt` Adversary does not use feedback from target model for attacks
`topK` Adversary has access to model's top-K prediction vector
`DistCheck` Distribution Check
2PC 2-party Computation

DPSGD Differentially Private Stochastic Gradient Descent

EU European Union

FL Federated learning

GPU Graphical Processing Unit

ML Machine learning
MPC Multi-party Computation

REE Rich Execution Environment

SGX Software Guard Extensions

TA Trusted Applications
TEE Trusted Execution Environment

ZKP Zero-Knowledge Proofs

List of Symbols

ω_{crpt}^{comm} Communication cost for cryptographic attestation
ω_{crpt}^{comp} Computation cost for cryptographic attestation
ω_{crpt} Cost for cryptographic attestation
ω_{inf} Cost for ML-based attestation
$\mathcal{N}_{acc}$ Number of accepted provers
$\mathcal{N}_{rej}$ Number of rejected provers
$\mathcal{N}_{spchk}$ Number of spotchecks
$\mathcal{N}_{spchk}$ Spotchecks for hybrid property attestation mechanism
$\mathcal{N}$ Total number of models
$\mathbb{P}_{crpt}$ Probability of fallback to cryptographic attestation
$\mathbb{P}_{inf}$ Probability of using ML-based attestation
$\mathbb{P}_{spchk}$ Probability of spotchecks
$\overline{\texttt{role}}$ Role of adversary in ML pipeline
$\mathcal{A}dv$ Adversary
f_{atst} Attestation classifier
f_{att} Attack model
S Space of all possible attributes corresponding to all data records
$\overline{\texttt{cap}}$ Adversary's capability (honest-but-curious or malicious)
$\mathbb{D}$ Underlying data distribution
$\mathbb{D}_{\mathbf{p}}$ Underlying data distribution with property P to sample datasets
z Data record consisting of input and corresponding label
$\mathcal{M}^{dtwm}$ Model trained with dataset watermarks
$\mathcal{D}_{te}^{aux}$ Auxiliary testing dataset
$\mathcal{D}_{tr}^{aux}$ Auxiliary training dataset
$\mathcal{D}^{aux}$ Auxiliary dataset
$\mathcal{D}_{te}^{\mathbf{bd}}$ Backdoored dataset
δ_{dp} Probability where the privacy loss exceeds the privacy budget
δ Adversarial noise added to input to generate adversarial examples

$\mathcal{D}_{te}^{rob}$ Dataset with adversarial examples
$\mathcal{D}_{tr}^{\textbf{sub}}$ Subset of training dataset
$\mathcal{D}_{te}^{prv}$ Prover's test dataset
$\mathcal{D}_{te}$ Testing dataset
$\mathcal{D}_{tr}^{prv}$ Prover's training dataset
$\mathcal{D}_{tr}$ Training dataset
$\mathcal{D}_{val}$ Validation dataset
$\mathcal{D}_{te}^{\textbf{wmM}}$ Watermarked dataset (model watermarking)
$\mathcal{D}_{te}^{\textbf{wmD}}$ Watermarked dataset (data watermarking)
$\epsilon_{\textbf{dp}}$ Privacy budget for differential privacy
ϵ_{rob} Perturbation budget for adversarial examples
$\mathcal{M}^{gpfr}$ Model trained with group fairness constraints
γ Model explanations
$\mathcal{G}_{err}$ Generalization error: train accuracy minus test accuracy
X Space of all input features corresponding to all data records
x Input feature corresponding to a specific data record
$\mathcal{I}$ Input to model $\mathcal{M}$ and equivalent to x
$\overline{\texttt{i}}$ Adversary's interaction with the target model
$\overline{\texttt{k}_\texttt{d}}$ Adversary's knowledge about training dataset
$\overline{\texttt{k}_\texttt{m}}$ Adversary's knowledge about model
Y Space of all true labels corresponding to all data records
y True label for a specific data record
l Loss function
$\mathcal{P}rv_{mal}$ Malicious prover
$\mathcal{M}_{ar}$ Model architecture
$\mathcal{M}_{2pc}$ Model obtained by training using two-party computation
$\mathcal{M}_{\theta}$ ML model with parameters
$\mathcal{M}$ ML model
$\mathcal{M}^{psnrob}$ Model trained with defense to suppress poisons/backdoors
$\overline{\texttt{obj}}$ Adversary's attack objectives
$\overline{\texttt{opt}}$ Adversary's optimization to perform attacks
$\mathcal{O}$ Output from model and equivalent to $\mathcal{M}(x)$
θ Model parameters
$\phi_{\textbf{acc}}$ Test accuracy
$\phi_{\textbf{ASR}}$ Attack success rate
$\phi_{\textbf{dp}}$ Metric to evaluate differential privacy
$\phi_{\textbf{eqodds}}$ Equalized odds
$\phi_{\textbf{err}}$ Error in model explanations

$\phi_{\textbf{robacc}}$ Accuracy on dataset with adversarial examples
$\phi_{\textbf{rsd}}$ Rate of successful detection for dataset watermarking
$\phi_{\textbf{wmacc}}$ Accuracy of model on watermarked dataset
p_{req} Value of the required property
$\mathcal{M}^{dp}$ Model trained with differential privacy
$\mathbf{P}$ Distributional property of some attribute
$\mathcal{D}^{\mathcal{P}rv}$ Prover's entire data
$\mathcal{P}rv$ Prover
$\mathcal{M}^{evsnrob}$ Model trained with defense against evasion
s Sensitive attribute
$\mathcal{S}rv_1$ Server-1 in two-party computation
$\mathcal{S}rv_2$ Server-2 in two-party computation
$\mathcal{M}^{shdw}$ Models used for optimizing various privacy attacks
$\mathcal{M}^{sur}$ Surrogate model trained to steal target model's functionality
$\mathcal{T}$ Training configuration (e.g., objective function, training hyperparameters, etc.)
$\mathcal{D}^{vrf}_{\mathbf{P}}$ Dataset for verification
$\mathcal{D}^{\mathcal{V}er}$ Verifier's entire data
$\mathcal{D}^{\mathcal{V}er}_{te}$ Verifier's data for testing mechanism
$\mathcal{D}^{\mathcal{V}er}_{tr}$ Verifier's data for training mechanism
$\mathcal{V}er$ Verifier
$\mathcal{M}^{mdlwm}$ Model trained with model watermarks

Chapter 1

Introduction

Over the past few years, machine learning (ML) algorithms have demonstrated substantial improvements across a wide range of tasks, including image classification [13, 14], natural language processing [15], as well as text and image generation [16]. These utility improvements have led to wide-scale deployment of ML models to various domains (e.g., education [17], healthcare [18, 19], finance [20, 21, 22], and businesses [23, 24]). This also includes various applications such as ***client-facing services*** (e.g., chatbots [25, 26], search engines [27, 28] and browsers [29, 30]), ***agentic systems*** [31, 32, 33], ***high-stakes applications*** (e.g., healthcare [34, 35, 36], predictive policing and granting paroles [37, 38, 39], and job applicant screening [40, 41]); and ***as components of larger systems*** (e.g., operating systems [42, 43] and autonomous vehicles [44, 45]).

Given the above wide-scale deployment of ML models, practitioners need to account for various concerns when deploying these models, including (i) ***infrastructure issues*** (e.g., latency, throughput, interoperability, scalability) [46], (ii) ***model design*** (e.g., high utility and generalization, minimal overfitting, hyperparameter tuning, data processing) [47, 48], (iii) ***environmental impact*** (e.g., reducing carbon emissions, water and power consumption by data centers), (iv) ***adversarial and societal risks*** (e.g., security, privacy, safety, unfairness, poor transparency, misalignment, misinformation, and cyberattacks), and (v) ***enabling governance*** (e.g., verifying practitioners' claims, and regulatory compliance).

Among the above deployment concerns, I focus on two, namely, *adversarial and societal risks*, and *enabling governance*. I will discuss one specific aspect within each deployment concern respectively, as separate parts of my thesis:

Part-1 (Unintended Interactions in Machine Learning): Within *adversarial and societal risks*, prior works have identified several risks to security, privacy, fairness, transparency, and

safety [49, 50, 6, 7, 10, 51, 11]. While substantial prior work explores the design of defenses against individual risks [52, 53, 54, 55, 56, 57], I argue that this is not sufficient for real-world deployment of ML models where practitioners must protect against multiple risks simultaneously. Thus, practitioners[1] need to address additional challenges beyond mitigating individual risks, including *unintended interactions* (includes trade-offs and unintended behavior stemming from ML model design choices in the presence of multiple adversaries with different objectives). A systematic understanding of these interactions is lacking, and I identify and explore three such interactions that are briefly described in Section 1.1 with more details in **Part-1**.

Part-2 (Accountability in Machine Learning Pipelines): Despite claiming to design and deploy effective defenses against individual risks and accounting for unintended interactions (from **Part-1**), we need technical mechanisms to hold the practitioners accountable for their claims. Such mechanisms can additionally support governance by enabling external verification that deployment practices comply with applicable regulations and policies. Thus, within the deployment concern of *enabling governance*, I study technical mechanisms that allow provers (e.g., model trainers) to demonstrate properties of an *ML model* (e.g., accuracy, fairness, robustness), *its training process* (e.g., how the model was trained and what it was trained on), and *its training dataset* (e.g., attribute distribution), to a verifier (e.g., regulator or customer). I refer to these as *ML property attestation mechanisms*, which can enable accountability across the ML pipeline, and support regulatory compliance. In **Part-2**, I systematically explore the design space of attestation mechanisms for ML properties, spanning both software- and hardware-based mechanisms, against the requirements of *effectiveness* (accurately measure properties), *efficiency* (incur minimal overhead relative to ML operations), *versatility* (cover diverse properties), *scalability* (can be verified by multiple verifiers), and *robustness* (resistant to evasion by malicious provers). I also discuss how these attestations can be extended to demonstrate which defenses were applied and how unintended interactions (from **Part-1**) were evaluated and addressed prior to deployment.

1.1 Part-1: Unintended Interactions in Machine Learning

I identify and systematically explore three types of *unintended interactions*: (a) a defense against one risk may increase or decrease other unrelated risks; (b) conflicts among defenses can decrease their effectiveness when combined; and (c) potential for collusion among adversaries can enable executing an attack to amplify others. Here, I defend the following thesis statement:

[1]**Terminology:** I use *practitioners* to broadly describe model designers, trainers, and service provers who design, train, and deploy ML models for various applications. I differentiate their roles later in Chapter 2: Section 2.1.

Thesis Statement

Unintended interactions can be systematically understood and predicted by identifying the underlying factors that influence the effectiveness of defenses and susceptibility to risks.

I will defend this thesis statement by addressing three research questions (**RQ1.1–RQ1.3**), each corresponding to the unintended interactions described above. I present an overview of the unintended interactions, corresponding research questions, and my contributions in addressing them in support of **Part-1**'s thesis statement:

- **Unintended Interactions among ML Defenses and Unrelated Risks:** Defenses targeting one risk can inadvertently increase or decrease susceptibility to other *unrelated risks*. For example, adversarial training (against the risk of evasion) may increase susceptibility to membership inference [58, 59], discriminatory behavior [60, 61], and makes it easy for an adversary to steal the target model's functionality [62]. However, a systematic understanding of the underlying causes of such interactions is lacking in the literature. Identifying these causes can enable practitioners to predict unexplored interactions *without expensive empirical evaluation*. However, this is challenging due to the opaque behavior of ML models. This raises the following research question:

  RQ1.1: *What are the factors underlying unintended interactions among defenses and unrelated risks, and how can we effectively predict unexplored ones?*

  I developed the *first* framework to systematically explore unintended interactions based on the conjecture that overfitting and memorization are the primary causes. This is because different risks depend on factors that influence overfitting and memorization, and an effective defense that changes these underlying factors, in turn affects the susceptibility to the risks. My framework includes various factors that influence overfitting and memorization: characteristics of the training dataset (e.g., size, distribution), the model (e.g., capacity), and the objective function (e.g., distance to boundary, loss curvature). Using this framework, I propose a guideline to explain interactions previously explored in the literature and conjecture about unexplored ones by examining how a defense changes a factor, and how a change in that factor correlates with a change in susceptibility to an unrelated risk. Using this guideline, I conjecture three previously unexplored interactions and empirically validate them. This work was published in IEEE Symposium on Security and Privacy (S&P) 2024, and received a *distinguished paper award* [1]. This also resulted in `Amulet`, an open-source library for evaluating unintended interactions [63], which *received funding from Intel*.
- **Combining ML Defenses without Conflicts:** Real-world models must simultaneously protect against multiple risks to security, privacy, and fairness. Combining existing defenses is plausible, but conflicting objectives can reduce the effectiveness of the constituent de-

fenses. This leads to asking the following:

RQ1.2: *How can we effectively combine ML defenses while avoiding conflicts?*

We need an easy-to-use *combination technique* that should be (i) *accurate* (correctly identifies whether a combination is effective), (ii) *scalable* (allows two or more defenses to be combined), (iii) *non-invasive* (requires no changes to the defenses being combined), and (iv) *general* (applicable to different types of defenses). I propose `Def\Con`, the *first technique* to effectively combine defenses based on whether the defenses will conflict with each other [2]. `Def\Con` identifies potential conflicts based on where each defense is applied in the ML pipeline (pre-, in-, or post-training), and whether the underlying mechanisms of the defenses interfere (e.g., a later defense minimizes a risk used by an earlier defense). Since these steps do not depend on specific defenses, `Def\Con` is generally applicable to various defenses. This helps practitioners identify compatible defenses that can be effectively combined, i.e., their effectiveness remains comparable to applying each defense separately. I show that `Def\Con` is *more accurate* than prior baselines, and correctly predicts seven out of the eight previously explored combinations, and 27 out of 30 unexplored combinations that my co-authors and I evaluated empirically. I also show, for the first time in the literature, that it is possible to effectively combine three defenses (i.e., `Def\Con` is scalable). Finally, `Def\Con` combines *existing defenses* without modification (i.e., `Def\Con` is non-invasive). This work was published in the Transactions on Machine Learning Research (TMLR), 2025.

- **Colluding Adversaries in ML Pipelines:** So far, while answering **RQ1.1** and **RQ1.2**, I assumed that adversaries are independent actors in the ML pipeline with distinct objectives targeting different risks. However, practitioners must consider the potential for *collusion* among adversaries within the ML pipeline, where an adversary executes an attack (by exploiting the vulnerability related to the corresponding risk) to amplify the effectiveness of attacks by other adversaries. Existing work lacks a systematic framework to explore potentially colluding adversaries, and to study the implications of the adversaries' characteristics for collusion. A unified framework can help practitioners and researchers (i) identify new threats against ML systems, (ii) design strong attacks for auditing, and (iii) design effective defenses against such attacks. This raises the following question:

RQ1.3: *What are the underlying factors that that aid the potential for collusion among adversaries, and how can we predict the collusion potential?*

I present the *first framework* to explore two collusion types: (i) between train- and inference--time adversaries (or "*train-inference collusion*", and (ii) among inference-time adversaries (or "*inference-time collusion*") [3]. In both cases, I observe that a successful attack changes cer-

tain factors, which can influence the effectiveness of other attacks relying on those factors. However, these factors differ between both collusion types. For *train-inference collusion*, poisoning changes the factors underlying overfitting and memorization which can impact the effectiveness of inference-time attacks that rely on those factors. Here, the framework includes the factors underlying overfitting and memorization (from Chapter 3), which can be manipulated by the train-time adversary through poisoning. For *inference-time collusion*, knowledge inferred from the first attack is treated as an outcome that satisfies the prerequisites for the second attack. Here, factors related to adversary's knowledge (e.g., of training data, target model) are part of the framework. Using the common factors among attacks, I present a guideline to predict the potential for collusion. I empirically validate the guideline on five unexplored cases of collusion, and discuss the impact of the adversary's characteristics. This work was published in the USENIX Security Symposium (USENIX SEC), 2026.

1.2 Part-2: Accountability in Machine Learning Pipelines

Having identified and explored the three unintended interactions in **Part-1**, we still need mechanisms to validate practitioners' claims regarding ML properties including defense deployment, the handling of unintended interactions, and the execution of standard ML operations throughout the ML pipeline. Practitioners can make claims about various ML properties using (i) *datasheets* to describe the training dataset properties (e.g., bias and distribution) [64], and (ii) *model cards* to describe how the model was trained and evaluated (e.g., accuracy, fairness, privacy, and robustness) [65]. Since there are no approaches to describe inference-time properties (e.g., use of filters or transforming the inputs and outputs, and binding the output from a model to the corresponding input and model), I propose *inference cards* [5]. Collectively, I refer to datasheets, model cards, and inference cards as *ML property cards*, which improve transparency across the ML pipeline. However, these ML property cards are *not verifiable* and allow malicious practitioners to make false claims about the model properties (e.g., Hugging Face where the descriptions of the uploaded models and datasets can be false [66]).

To prevent the inclusion of false claims in the ML property cards, there is a need for technical mechanisms that allow practitioners (as provers) to provide *verifiable claims* to a verifier (e.g., a regulator). These verifiable claims can be achieved through attestations which include (i) *proof of attribute distribution* (value of distribution of attributes in the training dataset), (ii) *proof of training* (model is trained on a specified dataset, configuration, and architecture), (iii) *proof of evaluation* (model has certain accuracy, fairness, or robustness on a given test dataset), and (iv) *proof of inference* (model generated an output for a given input). These mechanisms can help with regulatory compliance when verifiers check the attestations against ex-

pected behavior. However, they will differ from traditional approaches where a regulatory agency inspects and issues certificates. This is because (i) it may not be possible to disclose models and datasets due to intellectual property or privacy constraints (e.g., GDPR- or HIPAA-protected data), and (ii) companies might not wait for slow compliance checks in a competitive environment. Hence, there is a need for mechanisms to allow provers to *proactively demonstrate* their claims about ML properties to a verifier, while *preserving the model and data confidentiality*. In **Part-2**, I will defend the following thesis statement:

Thesis Statement

Ensuring accountability of ML operations by designing technical mechanisms that are effective, efficient, versatile, scalable, and robust, is feasible.

I support this claim by addressing research questions: **RQ2.1** and **RQ2.2**. I now present an overview of these questions and my contributions in answering them.

- **Software-based ML Property Attestation Mechanisms:**[2] I propose *ML property attestation mechanisms* that enable a prover (e.g., model trainer) to demonstrate relevant properties of a model, training dataset, and training process to a verifier (e.g., regulator) [4]. A natural first step is to answer the following research question:

  RQ2.1: *Can software-based ML property attestation mechanisms meet all requirements?*

  I use the case study of *distributional equity*, which uses *proofs of attribute distribution* to verify whether a training dataset's distribution for a given attribute (e.g., race or sex) matches a required distribution specified by policies or regulations. I explore three software-based mechanisms in this case study: (i) *ML-based mechanism* that re-purposes distribution inference attacks or statistical techniques to audit target ML models; (ii) *cryptographic mechanism* that uses protocols based on zero-knowledge proofs or multi-party computation; and (iii) *hybrid mechanism* that combines both ML-based and cryptographic mechanisms. I show that ML-based mechanisms are *neither effective nor robust*, while cryptographic mechanisms are *costly for practical deployment* [4]. I propose a hybrid mechanism that strategically combines ML-based and cryptographic attestations (using multi-party computation), to get *better* efficiency than cryptographic mechanisms and *better* effectiveness than ML-based mechanisms. However, it does not scale to multiple verifiers limiting its practicality in real-world settings. Hence, other attestation mechanisms need to be considered. This work was published in the European Symposium on Research in Computer Security (ESORICS), 2024.
- **Hardware-based ML Property Attestation Mechanisms:** In light of the limitations of

[2] **Terminology:** For brevity, I use "attestation mechanisms" in place of "ML property attestation mechanisms," and refer to the types as software-based, hardware-based, ML-based, cryptographic, and hybrid (described later).

the software-based mechanisms, a natural question is whether hardware-based mechanisms using trusted execution environments (TEEs) can be a practical alternative. TEEs have been prevalent in various devices (e.g., mobile phones) [67, 68], and recent advancements allow efficiently executing ML workloads (e.g., Intel's AMX, NVIDIA's H100 GPUs) [69]. This leads to the following research question:

RQ2.2: *Can hardware-based ML property attestation mechanisms address the limitations of software-based ones, and meet all requirements?*

I propose `Laminator`, a framework to provide ML property attestations using hardware-based TEEs (e.g., Intel SGX) [5]. I show that `Laminator` satisfies all required properties and can efficiently furnish verifiable ML property attestations. When combined with ML property cards, `Laminator`'s attestations enable practitioners to make provable claims about their datasets, models, inference-time behavior, and training processes. This can enable accountability in ML pipelines and also demonstrate compliance against various regulatory and policy requirements. This work was published in the ACM Conference on Data and Application Security and Privacy (CODASPY), 2025. The hardware-based ML property attestations provided by `Laminator` can be extended to demonstrate to a verifier that the unintended interactions from **Part-1** have been properly accounted for (discussed in Chapter 8).

I outline the thesis structure, listing chapters, research questions ("RQs"), contribution references ("Ref."), and publication statuses (Table 1.1).

Table 1.1: **Overview of chapters and their titles, corresponding research questions ("RQs"), references ("Ref."), and their statuses.**

Chp	RQs	Chapter Title	Ref.	Status
		Part-1: Unintended Interactions in ML		
3	RQ1.1	Unintended Interactions among ML Defenses and Unrelated Risks	[1]	IEEE S&P'24 🏆 [1]
4	RQ1.2	Combining ML Defenses without Conflicts	[2]	TMLR'25
5	RQ1.3	Colluding Adversaries in ML Pipelines	[3]	USENIX SEC'26
		Part-2: Accountability in ML Pipelines		
7	RQ2.1	Software-based ML Property Attestation Mechanisms	[4]	ESORICS'24
8	RQ2.2	Hardware-based ML Property Attestation Mechanisms	[5]	ACM CODASPY'25

[1] Distinguished Paper Award

Part 1
Unintended Interactions in Machine Learning

Substantial prior work has explored the design of defenses against individual risks. I argue that this is insufficient for the real-world deployment of ML models where practitioners must protect against multiple risks simultaneously. This introduces additional challenges, including *unintended interactions in ML*. I identify and explore three types of *unintended interactions*: (a) a defense against one risk may increase or decrease other, unrelated risks; (b) conflicts among defenses may reduce their effectiveness when combined; and (c) collusion among adversaries may enable the exploitation of one risk to amplify others. In **Part-1**, I defend the following:

Thesis Statement

Unintended interactions can be systematically understood and predicted by identifying the underlying factors that influence the effectiveness of defenses and susceptibility to risks.

I first present background on the relevant concepts (Chapter 2), and then systematically explore the three *unintended interactions* in separate chapters, each addressing the corresponding research questions to support the thesis statement: **(RQ1.1)** *What are the factors underlying unintended interactions among defenses and unrelated risks, and how can we effectively predict unexplored ones?* (Chapter 3) **(RQ1.2)** *How can we effectively combine ML defenses while avoiding conflicts?* (Chapter 4) **(RQ1.3)** *What are the underlying factors that aid the potential for collusion among adversaries, and how can we predict the collusion potential?* (Chapter 5)

Chapter 2

Background

I present an overview of the ML pipeline and the notations used throughout the thesis (Section 2.1). I then describe overfitting and memorization in ML (Section 2.2), various risks to ML models by presenting a streamlined threat model (Section 2.3), and finally, present the defenses to mitigate various risks (Section 2.4).

2.1 Machine Learning Pipeline and Notations

The ML pipeline involves multiple actors with distinct roles across three stages: *pre-training* (e.g., data processing), *in-training* (e.g., optimizing the model), and *post-training* (*evaluation* on test dataset, and *deployment*). Figure 2.1 is an overview of this pipeline assuming the model is deployed as a service that returns responses to a client's queries. I later discuss how this pipeline extends to other settings. Below, I discuss the various actors and their roles in the ML pipeline across the different stages.

Pre-training

Multiple data owners (`DtOwnr`; ❶) provide raw data that includes either their personal data or data collected from external sources such as the Internet or software repositories. A data provider (`DtProv`; ❷, ❼) then aggregates the data from `DtOwnr` and processes them to produce training dataset $\mathcal{D}_{tr}$ (for training $\mathcal{M}$), validation dataset $\mathcal{D}_{val}$ (for hyperparameter tuning), and a test dataset $\mathcal{D}_{te}$ (for evaluating different metrics). These datasets are sampled from an underlying distribution $\mathbb{D}_{\text{P}}$ satisfying some distributional property P (e.g., some ratio of females

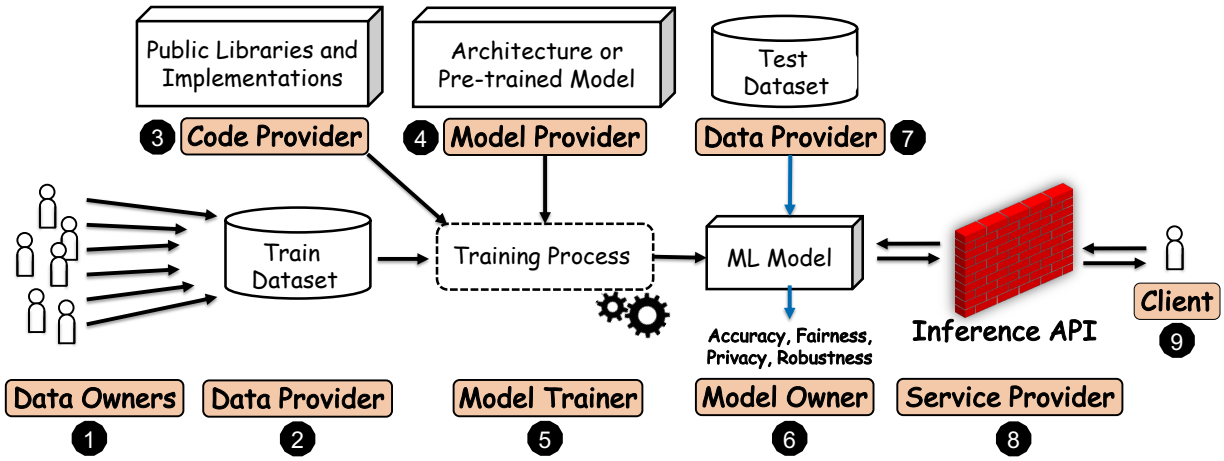


Figure 2.1: **Overview of ML pipeline**: Raw data from data owners (DtOwnr) is aggregated by a data provider (DtProv) and processed into training ($\mathcal{D}_{tr}$), validation ($\mathcal{D}_{val}$), and test ($\mathcal{D}_{te}$) datasets. A model trainer (ModTrnr) then trains or fine-tunes a model $\mathcal{M}$ using an architecture or pre-trained model from a model provider (ModProv), code for the training algorithm from a code provider (CodeProv), and a training configuration specified by ModTrnr. The trained $\mathcal{M}$, owned by a model owner (ModOwnr), is evaluated on $\mathcal{D}_{te}$ and subsequently deployed by a service provider (SrvProv) for clients (Clnt) to send queries and get responses.

in the distribution). Since i.i.d. sampling is used to get $\mathcal{D}_{tr}$ from $\mathbb{D}_{\text{P}}$ given by $\mathcal{D}_{tr} \sim \mathbb{D}_{\text{P}}$, $\mathcal{D}_{tr}$ also satisfies the property P. I consider datasets of the form (x, y), where $x \in X$ denotes an input record x from the input space X (e.g., images or text prompts), and $y \in Y$ denotes the corresponding expected output y from the space of all possible expected outputs Y (e.g., ground truth class for classifiers, or next-tokens for generative models). Here, the data record is represented as $z = (x, y)$. Additionally, the datasets might include an additional sensitive attribute s (e.g., race or sex) where $s \in S$ where S is the space of sensitive attributes. In this case, the data record is represented as $z = (x, y, s)$.

In-training

An ML model ($\mathcal{M}_\theta$[1]) is a function parameterized by θ that takes an input x (e.g., images, tabular data, graphs, or text prompts) and produces $\mathcal{M}(x)$ which is a classification label in case of classifiers, and an image or text for generative models. A model trainer (ModTrnr; ❺) trains $\mathcal{M}$

[1]Hereafter, I denote $\mathcal{M}_\theta$ by simply writing $\mathcal{M}$.

using the following training inputs:

- training configuration $\mathcal{T}$ provided by `ModTrnr` which includes the objective function, learning rate, weight decay, and other hyperparameters such as number of epochs;
- code for the training algorithm from a code provider (`CodeProv`; ❸) which includes third-party libraries (e.g., PyTorch) and public implementations (e.g., from GitHub repositories);
- model architecture (for training $\mathcal{M}$ from scratch) or a pre-trained model (for fine-tuning) from a model provider (`ModProv`; ❹) such as Hugging Face; and
- the training dataset $\mathcal{D}_{tr}$ from `DtProv`.

The trained model is obtained by applying the training process on the above inputs, which can be written as: $\mathcal{M} \leftarrow \text{Train}(\mathcal{D}_{tr}, \mathcal{M}_{ar}, \mathcal{T})$, where the code for Train() is obtained from `CodeProv`. During training, θ are updated iteratively to minimize the loss between $\mathcal{M}(x)$ and the expected output y. Specifically, we update θ using $(x, y) \in \mathcal{D}_{tr}$ over multiple epochs to minimize some objective function given by

$$\mathcal{C} : \min_{\theta} \; l(\mathcal{M}(x), y; \theta) + \lambda \, R(\theta)$$

where $l(\mathcal{M}(x), y)$ is the error on x given the ground truth y (e.g., expected classification label, image, or text). $R(\theta)$ is the regularization function that restricts θ from taking large values to reduce overfitting (described in Section 2.2), and λ is a hyperparameter to control regularization. The parameters θ are updated as: $\theta := \theta - lr.\frac{\partial \mathcal{C}}{\partial \theta}$ where lr is the learning rate.

Post-Training

During post-training, $\mathcal{M}$ is *evaluated* on $\mathcal{D}_{te}$ using various metrics (e.g., accuracy, robustness, fairness), followed by *deployment* to some application (e.g., ML-as-a-service or MLaaS).

Evaluation: I assume $\mathcal{M}$ is owned by a distinct actor in the ML pipeline, namely, model owner (`ModOwnr`; ❻). For instance, this is the setting when the training is outsourced by `ModOwnr` to `ModTrnr`. Once $\mathcal{M}$ is trained, `ModOwnr` can evaluate its utility on $\mathcal{D}_{te}$. The utility includes measuring accuracy in the case of classifiers, and the quality of generated images (e.g., using Fréchet Inception Distance) and text (e.g., using perplexity) in the case of generative models. Since the evaluation in the thesis primarily focuses on classifiers, I indicate the utility of $\mathcal{M}$ using its accuracy (ϕ_{acc}) on $\mathcal{D}_{te}$ which is computed as

$$\phi_{\text{acc}}(\mathcal{M}, \mathcal{D}_{te}) = \frac{1}{|\mathcal{D}_{te}|} \sum_{(x,y) \in \mathcal{D}_{te}} \mathbb{I}\{\mathcal{M}(x) = y\}$$

where $\mathcal{M}(x)$ is the most likely class.

Deployment: Assuming that the metrics evaluated on $\mathcal{D}_{te}$ are deemed good, `ModOwnr` shares $\mathcal{M}$ with a service provider (`SrvProv`; ❽), who deploys $\mathcal{M}$ as a service that allows clients (`Clnt`; ❾) to submit queries via an API and receive corresponding outputs. In other words, $\mathcal{M}$ takes an input $\mathcal{I}$ and produces output $\mathcal{O}$, i.e., $\mathcal{O} \leftarrow \mathcal{M}(\mathcal{I})$, where x is $\mathcal{I}$, and the output $\mathcal{M}(x)$ is $\mathcal{O}$. In case of open-weight models, `Clnt` can download the trained $\mathcal{M}$ (e.g., from Hugging Face) and query it locally. I refer to the output from $\mathcal{M}$, intermediate activations, θ, and gradients as *model observables*.

The pipeline in Figure 2.1 remains the same for other model types (e.g., generative models) and covers settings other than MLaaS. In federated learning (FL), participants act as both `DtOwnr` and `ModTrnr`, while the aggregator may serve as `SrvProv` and `ModOwnr` if they own $\mathcal{M}$. For agent-based systems, the pipeline additionally includes interactions between the agent and external Internet sources, where it retrieves information by invoking various tools and APIs based on instructions from `Clnt`.

2.2 Overfitting and Memorization in Machine Learning

Memorization and overfitting are related but distinct phenomena. I will illustrate the relationship between overfitting and memorization using ML classifiers[2].

Overfitting

ML classifiers are optimized to improve the *accuracy* on $\mathcal{D}_{tr}$ while ensuring that it generalizes to $\mathcal{D}_{te}$, i.e., high accuracy on both $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$. However, failing to generalize to $\mathcal{D}_{te}$ despite showing high accuracy on $\mathcal{D}_{tr}$ is ***overfitting***. We use *generalization error* ($\mathcal{G}_{err}$) to measure overfitting, which is the gap between the expected loss on $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ [71]:

$$\mathcal{G}_{err} = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{te}}[l(\mathcal{M}(x), y)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{tr}}[l(\mathcal{M}(x), y)]$$

Since a lower expected loss often leads to a higher accuracy, henceforth, I refer to $\mathcal{G}_{err}$ as the difference in accuracy between $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$.

[2]Distinction between overfitting and memorization for generative models can be found in Burg and Williams [70]

Memorization

We measure ***memorization***[3] with respect to a *single data record* in $\mathcal{D}_{tr}$. Memorization in classifiers can be categorized into the following:

- **Data Record Memorization:** For classifiers, $\mathcal{M}$ tends to memorize entire data records and their corresponding labels especially for outliers, mislabeled records, or those belonging to long-tail of $\mathcal{D}_{tr}$'s distribution [77, 78, 75]. This *label-only memorization* is essential for $\mathcal{M}$ to achieve high accuracy on $\mathcal{D}_{te}$ [74]. Furthermore, memorized data records exhibit a higher influence on model observables. We define *label-only memorization* of a data record $z_i = (x_i, y_i) \in \mathcal{D}_{tr}^{\text{sub}} \subseteq \mathcal{D}_{tr}$, as the difference in probability $\mathbb{P}(\mathcal{M}(x_i) = y_i)$ with and without z_i in $\mathcal{D}_{tr}$ [77, 79, 80]. This leave-one-out notion of memorization is given by

$$\texttt{mem}(\mathcal{A}, \mathcal{D}_{tr}^{\text{sub}}, i) := \mathbb{P}_{\mathcal{M} \leftarrow \mathcal{A}(\mathcal{D}_{tr}^{\text{sub}})}[\mathcal{M}(x_i) = y_i] - \mathbb{P}_{\mathcal{M} \leftarrow \mathcal{A}(\mathcal{D}_{tr}^{\text{sub}} \setminus z_i)}[\mathcal{M}(x_i) = y_i]$$

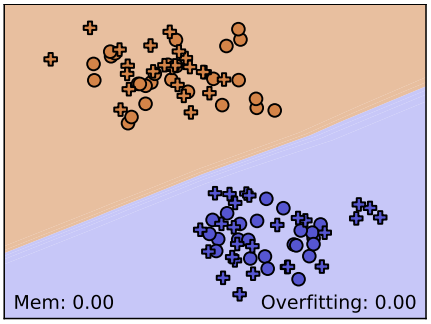
  If mem is close to one, it is likely that $\mathcal{M}$ has memorized z_i (i.e., not leave-one-out stable[4]). In other words, removing z_i from $\mathcal{D}_{tr}$ changes the decision boundary resulting in flipping $\mathcal{M}$'s prediction on z_i. On the other hand, mem=0 indicates no memorization suggesting leave-one-out stability. For generative models memorization of entire data records results in their verbatim replication as seen in language and text-to-image diffusion models [86, 87, 70]. This is influenced by the presence of duplicate data records in $\mathcal{D}_{tr}$ and generative model's large capacity [87, 72, 88]. Several metrics such as *exposure*, *k-Eidetic memorization*, and *counterfactual memorization* have been used for generative models [89].
- **Attribute Memorization:** $\mathcal{M}$ can selectively memorize specific attributes from $\mathcal{D}_{tr}$, which is essential for accurate predictions [90, 91]. There are two types of attributes that can be memorized by $\mathcal{M}$: *stable attributes*, which remain consistent despite changes in data distributions or *spurious attributes* which change $\mathcal{M}$'s prediction with change in distribution. $\mathcal{M}$'s generalization is better when it prioritizes learning stable attributes [90].

[3] **Terminology:** While "memorization" may carry negative connotations in ML, I use the term consistent with prior literature on ML risks (e.g., [70, 72, 73, 74, 75, 76]).

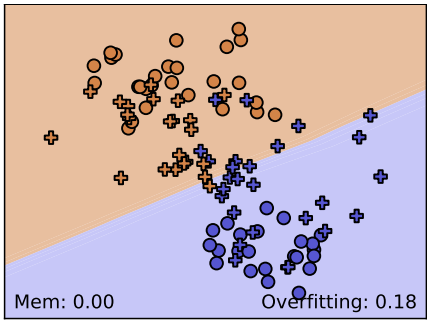
[4] **Related Work:** Although I adopt Feldman's notion of "label-only memorization" [77], the broader idea that sensitivity to individual training examples governs generalization has a long history in algorithmic stability [81, 82, 83, 84]. Stability characterizes generalization by limiting the influence of any single example, whereas memorization identifies examples whose influence is large, i.e., points on which the algorithm is highly unstable in the leave-one-out sense [85]. Feldman shows that such instability can be necessary for generalization on long-tailed distributions, refining the classical view that stability and generalization go hand in hand.

Relation between Overfitting and Memorization

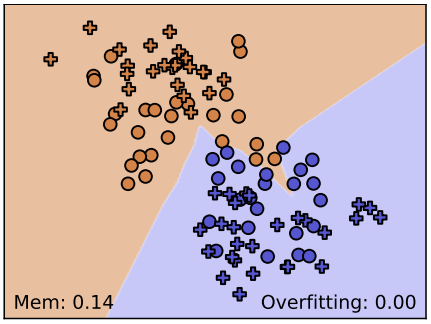
Figure 2.2 shows the four cases to indicate the relationship between overfitting and memorization in classifiers: (a) no overfitting and no memorization, (b) overfitting but no memorization, (c) no overfitting but memorization, and (d) both overfitting and memorization.



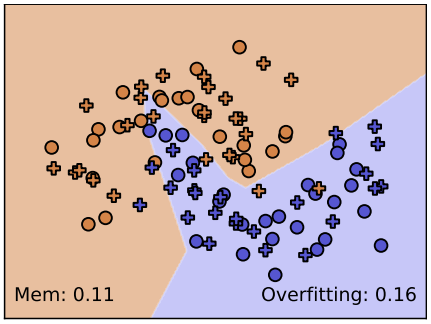
(a) No overfitting, no memorization



(b) Overfitting, no memorization



(c) Memorization, no overfitting



(d) Overfitting, memorization

Figure 2.2: **Relationship between overfitting and memorization:** Average `mem` across all data records in $\mathcal{D}_{tr}$ and overfitting ($\mathcal{G}_{err}$) are at the bottom. Circles indicate data records in $\mathcal{D}_{tr}$, and crosses indicate data records in $\mathcal{D}_{te}$.

Illustration Setup: I generate a synthetic dataset for binary classification and train a simple neural network with two hidden layers with ten neurons each. I measure overfitting using $\mathcal{G}_{err}$, and memorization by averaging `mem` across all data records in $\mathcal{D}_{tr}$.

Illustration: There are four possible cases depending on whether overfitting and memorization occurs in $\mathcal{D}_{tr}$. I highlight the cases below:

- **Figure 2.2 (a)** $\rightarrow$ ***No overfitting and no memorization:*** I generate the synthetic data such that the data records in $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ are linearly separable. Hence, $\mathcal{M}$ perfectly fits

$\mathcal{D}_{tr}$ and generalizes to $\mathcal{D}_{te}$ (i.e., $\mathcal{G}_{err} = 0$). Also, data records in $\mathcal{D}_{tr}$ are away from the decision boundary due to which there is no memorization (mem=0).

- **Figure 2.2 (b) → *Overfitting but no memorization*:** To illustrate overfitting without memorization in Figure 2.2 (b), I add noise to $\mathcal{D}_{te}$ in Figure 2.2 (a), causing some data records to cross the decision boundary. Here, $\mathcal{D}_{tr}$ is linearly separable but not $\mathcal{D}_{te}$. As a result, $\mathcal{M}$ perfectly fits $\mathcal{D}_{tr}$ but does not generalize to $\mathcal{D}_{te}$. Additionally, data records in $\mathcal{D}_{tr}$ are far from the decision boundary making them leave-one-out stable (mem=0).
- **Figure 2.2 (c) → *No overfitting but memorization:*** To illustrate memorization without overfitting in Figure 2.2 (c), I perturb data records in $\mathcal{D}_{tr}$ from Figure 2.2 (a) so that the data records move closer to the decision boundary, while keeping $\mathcal{D}_{te}$ linearly separable. The data records in $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ are separable resulting in no overfitting ($\mathcal{G}_{err} = 0$). However, with data records in $\mathcal{D}_{tr}$ being close to the decision boundary, there is memorization (mem > 0).
- **Figure 2.2 (d) → *Overfitting and memorization:*** To illustrate both overfitting and memorization in Figure 2.2 (d), I add noise to both $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ data records in Figure 2.2 (a). The noise shifts data records in $\mathcal{D}_{te}$ away from the distribution of $\mathcal{D}_{tr}$, inducing overfitting, while placing data records from different classes in $\mathcal{D}_{tr}$ close together, leading to their memorization. Here, $\mathcal{M}$ both overfits to $\mathcal{D}_{tr}$ (i.e., $\mathcal{M}$ fits $\mathcal{D}_{tr}$ perfectly but does not generalize to $\mathcal{D}_{te}$), and memorizes some data records in $\mathcal{D}_{tr}$ (i.e., not leave-one-out stable).

In real-world models where $\mathcal{D}_{tr}$ and $\mathcal{M}$ are complex, both overfitting and memorization typically occur together as seen in Figure 2.2 (d). Hence, hereafter I will assume this as the default.

2.3 Streamlined Threat Model for Machine Learning

Prior to deployment, practitioners need to design ML models to protect against various risks to security, privacy, fairness, and transparency. To describe the various risks, I present a streamlined threat model by defining the characteristics of the adversary ($\mathcal{A}dv$): objectives, knowledge and access to target model $\mathcal{M}$ and its training data $\mathcal{D}_{tr}$, and capabilities). Prior threat models in ML literature are *limited* to specific risks (e.g., [6, 7, 8, 11, 9, 10]); have *incomplete* characteristics (e.g., [92]); use *inconsistent terminology* (e.g., "blind" backdoors [93] vs. "blind" membership inference [94], or "no box" [95] vs. "blackbox" [6]); and do not *extend* to new risks, characteristics, and settings. The streamlined threat model presented below addresses the limitations of prior work by covering various risks with consistent terminology. Such a threat model can act as a community standard in ML, similar to other efforts in existing security and privacy literature (e.g., Dolev-Yao model in network security [96] and streamlining privacy terminology [97, 98]).

To define the threat model, I identify various $\mathcal{A}dv$'s characteristics (objectives, capabilities, knowledge and access to $\mathcal{M}$ and $\mathcal{D}_{tr}$). Formally, I represent the threat model as a tuple of seven sets (denoted as $\overline{<.>}$), corresponding to each of the questions below:

- what role(s) does $\mathcal{A}dv$ play? ($\overline{\texttt{role}}$);
- what are $\mathcal{A}dv$'s objectives? ($\overline{\texttt{obj}}$);
- what are $\mathcal{A}dv$'s capabilities? ($\overline{\texttt{cap}}$);
- what is $\mathcal{A}dv$'s interaction with $\mathcal{M}$? ($\overline{\texttt{i}}$)
- what attack optimizations can $\mathcal{A}dv$ use? ($\overline{\texttt{opt}}$)
- what is $\mathcal{A}dv$'s knowledge about $\mathcal{M}$? ($\overline{\texttt{k}_\texttt{m}}$);
- what is $\mathcal{A}dv$'s knowledge about data? ($\overline{\texttt{k}_\texttt{d}}$)

Given these characteristics, I denote a threat model ($\texttt{TM}$) describing $\mathcal{A}dv$ as a tuple of the above seven sets describing the characteristics:

$$\texttt{TM} = (\ \overline{\texttt{obj}},\ \overline{\texttt{role}},\ \overline{\texttt{cap}}, \overline{\texttt{opt}}, \overline{\texttt{k}_\texttt{m}},\ \overline{\texttt{i}},\ \overline{\texttt{k}_\texttt{d}}\ )$$

The elements of each of the seven sets constituting $\texttt{TM}$ are drawn from their respective spaces of all possible values. I denote these spaces with the first letter capitalized: $\overline{\texttt{Obj}}$ as the space for $\overline{\texttt{obj}}$, $\overline{\texttt{Role}}$ for $\overline{\texttt{role}}$, $\overline{\texttt{Cap}}$ for $\overline{\texttt{cap}}$, $\overline{\texttt{Opt}}$ for $\overline{\texttt{opt}}$, $\overline{\texttt{K}_\texttt{M}}$ for $\overline{\texttt{k}_\texttt{m}}$, $\overline{\texttt{I}}$ for $\overline{\texttt{i}}$, and $\overline{\texttt{K}_\texttt{D}}$ for $\overline{\texttt{k}_\texttt{d}}$. I describe the possible values in each space characterizing $\mathcal{A}dv$ and summarize them in Table 2.1.

$\mathcal{A}dv$'s Role indicates who $\mathcal{A}dv$ is in the ML pipeline (Figure 2.1). This in turn determines $\mathcal{A}dv$'s capabilities and their level of access to, and knowledge of $\mathcal{M}$ and $\mathcal{D}_{tr}$. This includes data owner ($\texttt{DtOwnr}$), data provider ($\texttt{DtProv}$), code provider ($\texttt{CodeProv}$), model provider ($\texttt{ModProv}$), model owner ($\texttt{ModOwnr}$), model trainer ($\texttt{ModTrnr}$), service provider ($\texttt{SrvProv}$), and client ($\texttt{Clnt}$).

$$\overline{\texttt{Role}} = \left\{ \begin{array}{l} \texttt{DtOwnr}, \texttt{DtProv}, \texttt{CodeProv}, \texttt{ModProv}, \\ \texttt{ModTrnr}, \texttt{ModOwnr}, \texttt{SrvProv}, \texttt{Clnt} \end{array} \right\}$$

$\mathcal{A}dv$'s Objectives includes various risks to security, privacy, and fairness, which $\mathcal{A}dv$ can exploit through different attacks[5]:

[5]**Terminology:** A risk arises from vulnerabilities that can be exploited by attacks. For example, $\texttt{Evsn}$, $\texttt{MemInf}$, $\texttt{AttInf}$, etc. represent risks stemming from different vulnerabilities (e.g., distinguishability in model observables across demographic groups). Each risk is associated with multiple attacks that exploit the underlying vulnerability (e.g., PGD [99] and FGSM [100] for $\texttt{Evsn}$). I refer to "an attack exploits a risk" for brevity.

Evasion (Evsn) modifies the inputs to force $\mathcal{M}$ to misclassify (for classifiers) [6, 7], or generate unacceptable images/text (for generative models) [101, 102]. In classifiers, these modified inputs are called adversarial examples ($x_{adv} = x + \delta$) obtained by adding perturbations δ to input x where δ is computed to maximize $\mathcal{M}$'s loss: $\max_{\|\delta\| \leq \epsilon_{rob}} \ell(\mathcal{M}(x + \delta), y)$. Here, δ is bounded within an l_p norm, i.e., $||\delta||_p < \epsilon_{rob}$ where ϵ_{rob} is the perturbation budget. Similarly, in generative models, adversarial prompts are generated to evade safeguards against unacceptable content (or jailbreaking).

Poisoning (Poisn) modifies $\mathcal{M}$ by training on malicious data records (or *poisons*) injected in $\mathcal{D}_{tr}$ to degrade $\mathcal{M}$'s utility on $\mathcal{D}_{te}$ [9]. A variant of poisoning is **backdoors (Bkdr)** where $\mathcal{M}$ is trained with poisons embedded with a specific pattern (aka *trigger*), which maps an input to $\mathcal{A}dv$-chosen output. Backdoored inputs will generate $\mathcal{A}dv$-chosen output, but clean inputs will result in normal behavior [103, 104, 105]. I refer to both of them as Poisn for brevity.

Unauthorized Model Ownership (ModExt) is when $\mathcal{A}dv$ derives a surrogate model ($\mathcal{M}^{sur}$) mimicking $\mathcal{M}$'s functionality. It may be an identical copy of $\mathcal{M}$ (*whitebox model theft* or physically stealing the model weights), or derived by querying $\mathcal{M}$ and using the corresponding outputs to train $\mathcal{M}^{sur}$ (*blackbox model theft or extraction*) [106, 107, 108, 109]. There are two types of $\mathcal{A}dv$: (a) *malicious suspects* who perform model theft but try to evade ownership verification by modifying $\mathcal{M}^{sur}$ to look different from $\mathcal{M}$, and (b) *malicious accusers* who falsely accuse an independently trained model as being $\mathcal{M}^{sur}$ [92]. I focus on the well-explored setting of malicious suspects who use blackbox model extrac-

Table 2.1: **Summary of the characteristics of $\mathcal{A}dv$ and their descriptions**, from which the elements of the sets constituting TM are drawn.

Category	Notation	Description
Role ($\overline{\texttt{Role}}$)	DtOwnr	Data Owner
	DtProv	Data Provider
	CodeProv	Code Provider
	ModProv	Model Provider
	ModTrnr	Model Trainer
	ModOwnr	Model Owner
	SrvProv	Service Provider
	Clnt	Client
Objective ($\overline{\texttt{Obj}}$)	Evsn	Evasion
	Poisn	Poisoning
	Bkdr	Backdoor
	MemInf	Membership Inference
	AttInf	Attribute Inference
	DistInf	Distribution Inference
	DtRecon	Data Reconstruction
	ModExt	Unauth. Model Ownership
	Discr	Discriminatory Behavior
Capability ($\overline{\texttt{Cap}}$)	HbC	Honest-but-Curious $\mathcal{A}dv$
	Mal	Malicious $\mathcal{A}dv$
Optimization ($\overline{\texttt{Opt}}$)	Adpt	Adaptive
	nonAdpt	Non-Adaptive
Model Knowledge ($\overline{\texttt{K}_\texttt{M}}$)	NoAccess	No Access
	BBox	Blackbox
	GBox	Graybox
	WBox	Whitebox
	hardLbl	Hard labels
	topK	Top-K Predictions
	fullPred	Full prediction vector
Interaction ($\overline{\texttt{I}}$)	NoAccess	No Access
	1Shot	One-Shot (Single Query)
	KShot	K-Shot (k Queries)
	Unlmtd	Unlimited Queries
Data Knowledge ($\overline{\texttt{K}_\texttt{D}}$)	NoAccess	No Access
	NoOvlap	No-overlap with $\mathcal{D}_{tr}$
	PartOvlap	Partial overlap with $\mathcal{D}_{tr}$
	FullOvlap	Full overlap with $\mathcal{D}_{tr}$
	NoBlind	Non-Blind
	Blind	Blind

tion [106, 107, 108, 109].

Unauthorized Data Usage (`UnauthData`) occurs when $\mathcal{M}$ is trained on datasets collected without consent (e.g., face images for facial recognition) [110, 111, 112]. This often arises from large-scale scraping or data sharing, thus, exposing individuals to privacy risks like profiling or identity inference, and potentially violating privacy laws.

Membership Inference (`MemInf`) identifies if an input was in $\mathcal{D}_{tr}$, by exploiting the distinguishability in model observables on data records inside and outside of $\mathcal{D}_{tr}$ [11, 113, 114]. This includes exploiting overconfident predictions, lower loss, distinct gradients, and higher confidence scores for data records in $\mathcal{D}_{tr}$. This poses a privacy risk when $\mathcal{D}_{tr}$ is sensitive, such as for individuals with health conditions.

Attribute Inference (`AttInf`) identifies the value of a sensitive attribute corresponding to an input, by exploiting the distinguishability in model observables across different attribute values [115]. To distinguish inputs with different attribute values, attacks exploit differences in prediction confidence, loss or gradients, and output distributions. In generative models, `AttInf` translates to identifying personally identifiable information [116]. `AttInf` poses a privacy risk as individuals may prefer to not disclose sensitive attributes (e.g., race) to avoid potential discrimination.

Data reconstruction (`DtRecon`) recovers inputs or data records from $\mathcal{D}_{tr}$ using access to $\mathcal{M}$ [117, 49, 87, 118, 86]. For generative models, $\mathcal{M}$ can generate outputs which either replicate or closely resemble confidential data records in $\mathcal{D}_{tr}$ [119, 86, 87]. For instance, in language models, data records in $\mathcal{D}_{tr}$ can be identified by ranking the generated data using multiple likelihood-based metrics, to identify generated data that are most likely memorized from $\mathcal{D}_{tr}$ [86].

Distribution Inference (`DistInf`) identifies the distributional properties of $\mathcal{D}_{tr}$ with respect to some attribute (e.g., race or sex) by exploiting the distinguishability in model observables when trained on datasets with different distribution values [120, 12, 121, 122, 123]. The attack assumes that a model trainer and $\mathcal{A}dv$ have access to $\mathbb{D}$ and sampling functions $\mathcal{G}_0$ and $\mathcal{G}_1$ that transform $\mathbb{D}$ to obtain a sub-distribution satisfying a particular property. For instance, $\mathcal{G}_0(\mathbb{D})$ can be 80% males and 20% females while $\mathcal{G}_1(\mathbb{D})$ can be 50% males and 50% females. $\mathcal{A}dv$ infers whether $\mathcal{M}$ was trained on the datasets sampled from these sub-distributions $\mathcal{G}_0(\mathbb{D})$ or $\mathcal{G}_1(\mathbb{D})$ (i.e., $\mathcal{D}_{tr}$ has 80% males or 50% males). `DistInf` exploits differences in predictions (e.g., confidence scores), loss values, gradients, or parameters that vary with the distribution values. This poses a confidentiality risk when inferred properties are sensitive and may enable subsequent harms (e.g., redlining).

Discriminatory Behavior (`Discr`)[6] is observed when $\mathcal{M}$ behaves differently across demo-

[6]Unlike other risks that involve an explicit $\mathcal{A}dv$, `Discr` does not assume $\mathcal{A}dv$ but I treat it like other risks by

graphic groups (e.g., race or gender) on $\mathcal{D}_{te}$ with respect to different metrics such as accuracy, false positive/negative rates [124, 55]. This arises when $\mathcal{D}_{tr}$ captures various societal biases which $\mathcal{M}$ learns during training. Identifying such biases is challenging as $\mathcal{M}$'s behaviour is *incomprehensible* (difficult to explain why $\mathcal{M}$ gave a specific output) [57].

$$\overline{\texttt{Obj}} = \left\{ \begin{array}{l} \texttt{Evsn}, \texttt{Poisn}, \texttt{Bkdr}, \texttt{MemInf}, \texttt{AttInf}, \\ \texttt{DistInf}, \texttt{DtRecon}, \texttt{ModExt}, \texttt{Discr} \end{array} \right\}$$

I further categorize the above risks as *train-time* (i.e., `Poisn` or `Bkdr`) and *inference-time* (i.e., `Evsn`, `MemInf`, `AttInf`, `DistInf`, `DtRecon`, `ModExt`, and `Discr`).

Capability of $\mathcal{A}dv$ indicates whether or not $\mathcal{A}dv$ deviates from the computation prescribed for their role. This includes **honest-but-curious** (`HbC`) or **malicious** (`Mal`), which have also been called "passive" and "active" respectively (e.g., [125, 120]). `HbC` cannot deviate from prescribed procedure, but can observe (and use) all visible data to exploit various risks. `Mal` can deviate to do additional arbitrary computations.

$$\overline{\texttt{Cap}} = \{\texttt{HbC}, \texttt{Mal}\}$$

Optimization indicates the type of optimization used by $\mathcal{A}dv$ for the attack. An **adaptive** $\mathcal{A}dv$ (`Adpt`) modifies their subsequent attack queries to $\mathcal{M}$, based on the feedback from $\mathcal{M}$'s previous outputs[7]. Alternatively, **non-adaptive** $\mathcal{A}dv$ (`nonAdpt`) use a fixed set of attack queries or optimize them locally without feedback from $\mathcal{M}$.

$$\overline{\texttt{Opt}} = \{\texttt{Adpt}, \texttt{nonAdpt}\}$$

Knowledge of Model includes **whitebox** (`WBox`), **blackbox** (`BBox`), **graybox** (`GBox`), and **no access** (`NoAccess`). For both `BBox` and `GBox`, the output can be **full predictions** (`fullPred`), **top-k predictions** (`topK`), or **hard labels** (`hardLbl`).

- **Whitebox** (`WBox`): $\mathcal{A}dv$ has full knowledge of $\mathcal{M}$'s architecture, parameters, training process and training configuration. For instance, $\mathcal{A}dv$ could have downloaded the pre-trained $\mathcal{M}$. Given access to $\mathcal{M}$'s parameters, $\mathcal{A}dv$ can compute gradients and intermediate activations.
- **Graybox** (`GBox`): $\mathcal{A}dv$ knows $\mathcal{M}_{ar}$ and the training configuration $\mathcal{T}$. However, $\mathcal{A}dv$ does not know $\mathcal{M}$'s parameters.

assuming the measurement of discriminatory behavior is done by $\mathcal{A}dv$, for uniformity.

[7] **Terminology:** ML literature misuses "adaptive" to describe $\mathcal{A}dv$ with knowledge of the defense [126, 127, 128, 8], whereas in security literature, such knowledge is assumed by default. I call $\mathcal{A}dv$ without this knowledge "naïve". Following standard security terminology [129, 130], I define "adaptive" $\mathcal{A}dv$ as one who updates the attacks based on previous responses, and assume that $\mathcal{A}dv$ knows the defense by default.

- **Blackbox** (BBox): $\mathcal{A}dv$ does not know $\mathcal{M}_{ar}$, parameters, training process and $\mathcal{T}$. $\mathcal{A}dv$ can only query $\mathcal{M}$ via an API and get responses.
- **No access** (NoAccess): $\mathcal{A}dv$ cannot query $\mathcal{M}$, and only has access to some outputs for previously pre-computed inputs.

For both BBox and GBox, the following cases apply: **Hard-Label** (hardLbl), **Top-K Predictions** (topK), and **Full Predictions** (fullPred). **Hard-label** (hardLbl) is where $\mathcal{A}dv$ only receives the final classification result for a given input, such as malware classifiers producing labels (i.e., malware or not). **Top-K predictions** (topK) is where $\mathcal{A}dv$ obtains the top-k confidence scores from the prediction vector ($1 \leq k \leq N$) with N classes (e.g., ImageNet models output scores for a subset of classes). Finally, **full predictions** (fullPred) is where $\mathcal{A}dv$ receives the confidence scores for all classes (e.g., in high-stakes settings like credit approval).

I use (BBox, GBox) $\times$ (hardLbl, topK, fullPred) to denote their six possible combinations. Additionally, WBox corresponds to fullPred (WBox.fullPred), which I omit for brevity.

$$\overline{\texttt{K}_\texttt{M}} = \{\texttt{NoAccess},\ (\texttt{BBox},\ \texttt{GBox}) \times (\texttt{hardLbl},\ \texttt{topK},\ \texttt{fullPred}), \texttt{WBox}\}$$

Interaction indicates how $\mathcal{A}dv$ interacts with $\mathcal{M}$ which includes the following cases: (i) **no access (NoAccess)**; (ii) **one-shot (1Shot)** where $\mathcal{A}dv$ can only query once (e.g., evading a malware classifier on a target machine); (iii) **k-Shot (KShot)** where $\mathcal{A}dv$ can send k queries to $\mathcal{M}$ (e.g., in case of budget constraints); (iv) **unlimited (Unlmtd)** where $\mathcal{A}dv$ has no limit on the number of queries to $\mathcal{M}$ (e.g., with local access to $\mathcal{M}$).

$$\overline{\texttt{I}} = \{\texttt{NoAccess}, \texttt{1Shot}, \texttt{KShot}, \texttt{Unlmtd}\}$$

Knowledge of Data indicates the assumptions about $\mathcal{A}dv$'s auxiliary dataset $\mathcal{D}^{aux}$ for designing and evaluating the attack, using the datasets $\mathcal{D}^{aux}_{tr}$ and $\mathcal{D}^{aux}_{te}$ respectively. For access to $\mathcal{D}_{tr}$ and overlap with $\mathcal{D}^{aux}$, I indicate whether $\mathcal{A}dv$ has **no access** (NoAccess), **no overlap** (NoOvlap), **partial overlap** (PartOvlap) or **full overlap** (FullOvlap). In case where $\mathcal{A}dv$ has access to $\mathcal{D}^{aux}_{tr}$ or $\mathcal{D}^{aux}_{te}$, I indicate if $\mathcal{A}dv$ has ground truth: **non-blind** (NoBlind) when available, and **blind** (Blind) otherwise (e.g., when labeling is expensive).

- **No Access** (NoAccess): Specific roles in the ML pipeline do not have access to $\mathcal{D}_{tr}$ (e.g., CodeProv or ModProv). Hence, $\mathcal{A}dv$ as one of these roles will not have access to $\mathcal{D}_{tr}$.
- **No Overlap** (NoOvlap): $\mathcal{A}dv$ knows $\mathcal{D}_{tr}$'s underlying distribution to sample $\mathcal{D}^{aux}$ with no overlap with $\mathcal{D}_{tr}$. Also, $\mathcal{D}^{aux}$ could include both natural and synthetic datasets.
- **Partial Overlap** (PartOvlap): $\mathcal{D}^{aux}$ has partial overlap with $\mathcal{D}_{tr}$. For instance, this is the setting when part of $\mathcal{D}_{tr}$ is publicly available for $\mathcal{A}dv$ to download and use.

- **Complete Overlap** (`FullOvlap`): $\mathcal{D}^{aux}$ is the same as $\mathcal{D}_{tr}$ in the case where $\mathcal{D}_{tr}$ is publicly available (e.g., OpenAI's GPT2 model).

For each of `NoOvlap`, `PartOvlap`, and `FullOvlap`, $\mathcal{A}dv$ can either have ground truth available (`NoBlind`) or not (`Blind`). I denote the possible combinations as {`NoOvlap`, `PartOvlap`, `FullOvlap`} $\times$ {`NoBlind`, `Blind`}.

$$\overline{\texttt{K}_\texttt{D}} = \{(\texttt{NoOvlap}, \texttt{PartOvlap}, \texttt{FullOvlap}) \times (\texttt{NoBlind}, \texttt{Blind})\}$$

2.4 Defenses for Machine Learning Models

I describe the defenses proposed in the literature to mitigate various risks. For each defense, I specify the defense variants during *pre-training* that transform $\mathcal{D}_{tr}$ ("<defense>.**`Pre`**"), *in-training* that modify the objective function ("<defense>.**`In`**"), and *post-training* that apply filters on inputs/outputs or pruning $\mathcal{M}$ ("<defense>.**`Post`**"). For additional context, I indicate the impact of applying a defense on ϕ_{acc} compared to a "no defense" baseline, where "$\vee$" is a decrease, "$\sim$" is no effect, and "$\wedge$" is an increase.

Evasion robustness (`EvsnRob`) refers to the defenses that protect against `Evsn`. I refer to the corresponding model as $\mathcal{M}^{evsnrob}$.

- **`EvsnRob-Pre` (Data Augmentation)** where transforming data records in $\mathcal{D}_{tr}$ improves robustness [131, 132, 133, 134]. For instance, CutMix replaces a random patch of one image with a patch from another and mixes their labels proportionally to improve generalization and robustness [131]. Such techniques improve ϕ_{acc} ($\wedge$) by providing a regularizing effect during training [131, 132, 133, 134].
- **`EvsnRob-In` (Adversarial Training)** modifies the objective function for training to minimize the maximum loss from adversarial examples [99, 135]:

$$L_{advtr} = \min_{\theta} \frac{1}{|\mathcal{D}_{tr}|} \sum_{x,y \in \mathcal{D}_{tr}} \max_{\|\delta\| \leq \epsilon_{rob}} \ell(\mathcal{M}(x+\delta), y)$$

  Alternatively, randomized smoothing modifies the training and inference to obtain certified robustness of $\mathcal{M}$ [136, 137]. These defenses decrease ϕ_{acc} ($\vee$) [135, 138].
- **`EvsnRob-Post` (Input Processing)** filters adversarial perturbations before passing them to $\mathcal{M}$ (e.g., using generative models [139, 140] or input encoding [141, 142, 143]) or checks for adversarial examples using statistical tests [144]. Defenses that modify input images using generative models decrease ϕ_{acc} ($\vee$) [139, 140, 142, 143]. If the input transformation is small, the decrease in ϕ_{acc} is negligible ($\sim$) [141, 144].

Poison robustness (`PoisnRob`) includes defenses against *poisoning* and *backdoors*, by treating them as outliers in $\mathcal{D}_{tr}$. I refer to the corresponding model with `PoisnRob` as $\mathcal{M}^{psnrob}$.

- **`PoisnRob-Pre` (Data Sanitization)** involves detecting and removing outliers from $\mathcal{D}_{tr}$. This includes using Shapley values [145, 146, 147] or other techniques for anomaly detection [148, 149, 150, 151, 152]), followed by retraining. As the outliers are memorized and contribute to ϕ_{acc}, their removal degrades ϕ_{acc} ($\vee$) [145, 146].
- **`PoisnRob-In` (Fine-tuning)** trains $\mathcal{M}$ to minimize the influence of outliers as part of the objective function. This includes distillation to reduce the influence of poisons [153] or fine-tuning on a poison-free dataset [154, 155, 156, 157, 158]. These do not impact ϕ_{acc} ($\sim$).
- **`PoisnRob-Post` (Pruning)** reduces the effectiveness of backdoors by pruning some model parameters (by setting them as zero) based on the observation that poison and clean data records have different model observables such as activations [159, 160, 161, 162, 163]. Excessive pruning of $\mathcal{M}$ to reduce the effectiveness of backdoors, also degrades ϕ_{acc} ($\vee$).

Model Watermarking (`MdlWM`) checks for *unauthorized model ownership* including *model extraction attacks*. The goal is to watermark $\mathcal{M}$ such that the watermarks transfer to $\mathcal{M}^{sur}$ after model extraction. I refer to the corresponding watermarked model as $\mathcal{M}^{mdlwm}$. If a suspect model's watermark accuracy is above some pre-defined threshold, it is identified as $\mathcal{M}^{sur}$.

- **`MdlWM-Pre` (Backdoors)** use backdoors (e.g., small patches applied on images) as watermarks in $\mathcal{D}_{tr}$ [164, 165, 166, 167]. When trained on these watermarks, $\mathcal{M}^{mdlwm}$ learns to map the data records with the watermarks to a specific output (e.g., classification label). The accuracy of a model on watermarked data records in $\mathcal{D}_{te}$, can help check if the model is watermarked. These watermarks are designed to be stealthy and maintain ϕ_{acc} ($\sim$).
- **`MdlWM-In` (Optimization)** updates the original objective function to train $\mathcal{M}$ to include watermark behavior (i.e., behave differently on specific data records in $\mathcal{D}_{tr}$) [168, 93]. For instance, backdoors can be introduced through regularization which can be repurposed as watermarks [93]. This degrades ϕ_{acc} ($\vee$).
- **`MdlWM-Post` (API)** modifies outputs from $\mathcal{M}$ to embed watermarks instead of adding watermarked data records to $\mathcal{D}_{tr}$ [169]. These outputs will be used by $\mathcal{A}dv$ to train $\mathcal{M}^{sur}$ which will in turn transfer the watermarks to $\mathcal{M}^{sur}$. This defense maintains ϕ_{acc} ($\sim$).

Fingerprinting (`Fngrprnt`) checks for *unauthorized model ownership* by generating unique identifiers or *fingerprints* (e.g., adversarial examples, embeddings) for $\mathcal{M}$. These fingerprints transfer from $\mathcal{M}$ to any $\mathcal{M}^{sur}$ that is derived from it, but are distinct from the fingerprints of independently trained models [170, 171, 172, 173, 174]. Unlike watermarks which are invasive, fingerprints do not require retraining or modification of $\mathcal{M}$. Hence, all fingerprints are post-training schemes (denoted as **`Fngrprnt-Post`**) and have no effect on ϕ_{acc} ($\sim$).

Data watermarking (`DtWM`) checks for *unauthorized data use* where $\mathcal{M}$ is trained on datasets

without consent (e.g., face images for facial recognition) [110, 111, 112]. DtWM either adds watermarks (e.g., backdoors) to $\mathcal{D}_{tr}$ [175, 110, 52], or selects high-influence data records from $\mathcal{D}_{tr}$ as watermarks [176]. During verification, statistical tests [110] or membership inference [176] can check whether watermarks were used in $\mathcal{D}_{tr}$. Currently, DtWM is in pre-training stage (**DtWM-Pre**), and designed to maintain ϕ_{acc} ($\sim$). The watermarked model is denoted as $\mathcal{M}^{dtwm}$. The difference between MdlWM and DtWM is how a model trained from scratch on $\mathcal{D}_{tr}$ is classified: DtWM flags it for unauthorized data use, while MdlWM classifies it as independently trained.

Differential privacy (DiffPriv) mitigates privacy risks like MemInf and DtRecon [177]. Given two models trained on neighboring datasets differing by one record, DiffPriv bounds the privacy loss (distinguishability in predictions between the two models) by $e^{\epsilon_{\text{dp}}} + \delta_{dp}$. Here, $e^{\epsilon_{\text{dp}}}$ is the privacy budget and δ_{dp} is the probability that the privacy loss is $> e^{\epsilon_{\text{dp}}}$. I refer to the corresponding model with DiffPriv as $\mathcal{M}^{dp}$.

- **DiffPriv-Pre (Private Data)** involves generating synthetic datasets with privacy constraints according to DiffPriv. $\mathcal{M}$ is then trained on the synthetic dataset instead of $\mathcal{D}_{tr}$ [178, 179, 180, 181]. Training $\mathcal{M}$ on synthetic dataset decreases ϕ_{acc} compared to training on $\mathcal{D}_{tr}$ ($\vee$).
- **DiffPriv-In (DPSGD)** adds carefully computed noise to the gradients of $\mathcal{M}$ during training, to minimize the influence of individual data records on the model observables [177]. Alternatively, private aggregation of teacher's ensembles (PATE) [182] is another framework for DiffPriv where multiple teacher models are trained on disjoint private datasets, while a student model is trained on a public dataset with the labels annotated via noisy voting from the teacher models. These defenses decrease ϕ_{acc} ($\vee$) [183].
- **DiffPriv-Post (Output Perturbation)** adds carefully computed noise to the outputs of an empirical risk minimization objective [184]. This decreases ϕ_{acc} ($\vee$). The theoretical guarantees are poorer than other defenses for DiffPriv, and requires the objective function to be convex, which does not apply for neural networks. Hence, I omit this in the thesis.

Group fairness (GpFair) minimizes *discriminatory behavior* by ensuring $\mathcal{M}$ behaves equitably across demographic groups (e.g., race or sex) [55, 124]. GpFair is measured using various metrics like accuracy parity, demographic parity [210], equalized odds and equality of opportunity [200]. I refer to the corresponding model with GpFair as $\mathcal{M}^{gpfr}$.

- **GpFair-Pre (Fair Data)** modifies $\mathcal{D}_{tr}$ by reweighting, relabeling, or transforming feature representations across demographic groups so that outcomes are more balanced [188, 189, 190, 191]. These transformations aim to reduce systematic group-level bias before model training, thereby ensuring group fairness in the downstream classifier. This degrades ϕ_{acc} ($\vee$).
- **GpFair-In (Regularization)** adds an additional regularization term which penalizes violations in fairness constraints with respect to a specific metric (e.g., demographic parity,

Table 2.2: **Summary of risks and defenses, and their impact on accuracy:** I indicate defenses at different stages of the ML pipeline (pre-, in-, post-training). Column "ϕ_{acc}" indicates the impact of the defense on ϕ_{acc} where "$\vee$" $\rightarrow$ decrease, "$\sim$" $\rightarrow$ no effect, and "$\wedge$" $\rightarrow$ increase.

Risk	Defense	ϕ_{acc}	References
Evasion (`Evsn`)	**`EvsnRob` (Evasion Robustness)**		
	• `EvsnRob-Pre` (Data Augmentation)	$\wedge$	[131, 132, 133, 99, 134]
	• `EvsnRob-In` (Adversarial Training)	$\vee$	[135, 136, 137]
	• `EvsnRob-Post` (Input Processing)	$\vee$	[139, 140, 142, 143]
		$\sim$	[144, 141]
Poison (`Poisn`)/ Backdoor (`Bkdr`)	**`PoisnRob` (Outlier Robustness)**		
	• `PoisnRob-Pre` (Data Augmentation)	$\vee$	[185, 186, 148, 149, 187, 145, 146]
	• `PoisnRob-In` (Fine-tuning)	$\sim$	[154, 156, 157, 158, 155, 159, 160, 153]
	• `PoisnRob-Post` (Pruning)	$\vee$	[161, 162, 163]
Unauthorized Model Ownership (`ModExt`)	**`MdlWM` (Watermarking-M)**		
	• `MdlWM-Pre` (Backdoors)	$\sim$	[164, 165, 166, 167]
	• `MdlWM-In` (Optimization)	$\vee$	[168]
	• `MdlWM-Post` (API-based)	$\sim$	[169]
	`Fngrprnt` (Fingerprinting)		
	• `Fngrprnt-Post` (Fingerprints)	$\sim$	[170, 171, 172, 173, 174]
Unauthorized Data Usage (`UnauthData`)	**`DtWM` (Watermarking-D)**		
	• `DtWM-Pre` (Backdoors)	$\sim$	[175, 110, 176]
Inference Attacks (`MemInf`, `AttInf`, `DistInf`, `DtRecon`)	**`DiffPriv` (Differential Privacy)**		
	• `DiffPriv-Pre` (Private Data)	$\vee$	[179, 180]
	• `DiffPriv-In` (DPSGD)	$\vee$	[177, 182]
Discriminatory Behavior (`Discr`)	**`GpFair` (Group Fairness)**		
	• `GpFair-Pre` (Fair Data)	$\vee$	[188, 189, 190, 191]
	• `GpFair-In` (Regularization)	$\vee$	[192, 193, 194, 195, 196, 197, 198]
	• `GpFair-Post` (Calibration)	$\vee$	[199, 200, 201, 202]
	`Expl` (Explanations)		
	• `Expl-Post` (Attributions)	$\sim$	[203, 204, 205, 206, 207, 208, 209]

equalized odds) [196, 195, 192, 198]. For instance, the objective function to train $\mathcal{M}$ while satisfying demographic parity is given by

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda_{\text{DemPar}} \widehat{\mathcal{R}}_{\text{DemPar}}$$

where λ_{DemPar} is the regularization hyperparameter and

$$\widehat{\mathcal{R}}_{\text{DemPar}} = \left| \frac{1}{n_0} \sum_{i:s_i=0} f_\theta(x_i) - \frac{1}{n_1} \sum_{i:s_i=1} f_\theta(x_i) \right|$$

Here, n_0 and n_1 are the number of data records with the sensitive attribute $s_i = 0$ and $s_i = 1$ respectively. These additional constraints degrade ϕ_{acc} ($\vee$) [211, 212, 213].

- **`GpFair-Post` (Calibration)** adjusts the threshold over the predictions (i.e., probability vector indicating the likelihood of each class) to ensure that the prediction probabilities accurately reflect the true likelihood across each demographic group [199, 200, 201, 202, 214, 215, 216]. This degrades ϕ_{acc} ($\vee$) [199].

Explanations (`Expl`) improves comprehension of $\mathcal{M}$'s behavior. *Attribution-based methods* assign a score to each input feature to show how much it contributes to $\mathcal{M}$'s prediction using a simple local linear approximation [203, 217, 205]. *Influence-based methods* identify the data records in $\mathcal{D}_{tr}$ that influenced $\mathcal{M}$'s prediction the most on an input [206, 218]. *Recourse-based (counterfactual) methods* provide an alternative data record that makes the smallest change to an input data record to obtain the desired outcome [207]. `Expl` can help detect *discriminatory behavior* [208, 209], and is applied during the post-training stage (**`Expl-Post`**). Hence, `Expl` requires no retraining and does not degrade ϕ_{acc} ($\sim$).

I summarize all the risks, corresponding defenses, and their impact on ϕ_{acc} in Table 2.2.

Chapter 3

Unintended Interactions among ML Defenses and Unrelated Risks

Overview

When a defense is effective in mitigating one risk, it may correspond to an increased or decreased susceptibility to *other unrelated risks*. Existing research lacks an effective framework to recognize and explain such *unintended interactions among defenses and unrelated risks*. I present such a framework, based on the conjecture that *overfitting* and *memorization* underlie these unintended interactions. I survey existing literature on such unintended interactions, and accommodate them within the framework. I then use the framework to conjecture about three previously unexplored interactions, and empirically validate them. Overall, I show that such unintended interactions can be understood and predicted by studying underlying factors that influence overfitting and memorization, thus, answering **RQ1.1** to support the thesis statement.

Several *risks* to the security, privacy, and fairness of ML models have been identified and exploited, followed by the design of *defenses* to mitigate those risks [52, 53, 54, 55, 56, 57]. When a defense is effective against a specific risk, it may correspond to an increased or decreased *susceptibility to other risks* [219, 220]. For instance, adversarial training to minimize evasion also increases susceptibility to *membership inference* [58, 59]. I refer to these as *unintended interactions* between defenses and unrelated risks.

Foreseeing such unintended interactions is challenging. A unified framework enumerating different defenses and risks can help (a) *researchers* to identify unexplored interactions and design algorithms with better trade-offs, and (b) *practitioners* to account for such interactions

before deployment. However, prior works are limited to specific risks (e.g., membership inference [220]), defenses (e.g., fairness with privacy risks [221]) or interactions (e.g., explanations with security risks [50]), and do not systematically study the underlying causes (e.g., Gittens et al. [219]). A comprehensive framework spanning multiple defenses and risks, to systematically identify potential unintended interactions is currently absent.

I fill this gap by systematically examining various unintended interactions spanning multiple defenses and risks, and answering the following research question: (**RQ1.1**) *what are the factors underlying unintended interactions among defenses and risks, and how can we effectively predict unexplored ones?* I conjecture that *overfitting* and *memorization* of the training data are the potential causes underlying these unintended interactions. An effective defense may induce, reduce or rely on overfitting or memorization, which in turn impacts the model's susceptibility to other risks. I identify various factors such as the characteristics of the training dataset, objective function, and the model, that collectively influence a model's propensity to overfit or memorize. These factors provide a nuanced understanding of the susceptibility to different risks when a defense is effective. I claim the following contributions:

1. the *first systematic framework* to understand unintended interactions by studying their underlying causes and factors that influence them; (Section 3.1)
2. a literature *survey* to identify various unintended interactions, situating them within the framework, and a guideline to conjecture about unintended interactions; (Section 3.2)
3. identifying *unexplored unintended interactions* for future research, using the guideline to *conjecture three such interactions*, and *empirically validate* them. (Section 3.3)

3.1 Framework to Explore Interactions

I introduce a framework to examine how defenses interact with various risks based on the conjecture that *overfitting* and *memorization* are the underlying causes affecting other risks (validated in Section 3.2). Recall the discussion on overfitting and memorization from Chapter 2: Section 2.2. I will now present the various factors influencing overfitting and memorization that are part of the proposed framework to explore unintended interactions.

Factors Influencing Overfitting

I present two factors that influence overfitting: (i) size of training data $\mathcal{D}_{tr}$, i.e., $|\mathcal{D}_{tr}|$ (D1), and (ii) capacity of model $\mathcal{M}$ (M1) [222, 223, 224, 225]. They can be understood from the perspective of bias and variance: bias is an error from poor hyperparameter choices for $\mathcal{M}$ and variance

is an error from sensitivity to changes in $\mathcal{D}_{tr}$ [222]. High bias can prevent $\mathcal{M}$ from learning relevant relationships between attributes and labels, while high variance leads to fitting noise in $\mathcal{D}_{tr}$. These can result in overfitting. There is a trade-off between bias and variance; ideally, one wants to find the right balance with the appropriate choice of D1 and M1.

D1 **Size of $\mathcal{D}_{tr}$ ($|\mathcal{D}_{tr}|$):** Increasing $|\mathcal{D}_{tr}|$ reduces overfitting because $\mathcal{M}$ is exposed to a broader and more representative sample of the underlying data distribution during training. As $|\mathcal{D}_{tr}|$ increases, it is more likely that new inputs during inference resemble data records that $\mathcal{M}$ has already learned from. This reduces $\mathcal{M}$'s reliance on the noise present in $\mathcal{D}_{tr}$. From a bias–variance perspective, a larger $\mathcal{D}_{tr}$ reduces variance: $\mathcal{M}$'s parameter estimates are more stable and less sensitive to fluctuations in any particular subset of $\mathcal{D}_{tr}$. This improves generalization and reduces overfitting.

M1 **$\mathcal{M}$'s capacity:** For a fixed size of $\mathcal{D}_{tr}$, as $\mathcal{M}$'s capacity increases, bias decreases but variance increases. If $\mathcal{M}$ becomes too large, it may overfit, whereas $\mathcal{M}$ with an reasonable capacity strikes a balance between bias and variance, and generalizes to new data.

Factors Influencing Memorization

As part of the framework, I also identify and include factors that influence memorization of data records in $\mathcal{D}_{tr}$. This includes factors related to the characteristics of $\mathcal{D}_{tr}$ such as (i) the tail length of $\mathcal{D}_{tr}$'s distribution (D2), (ii) the number of input attributes (D3), and (iii) $\mathcal{M}$'s priority of learning stable attributes (D4; recall from Chapter 2: Section 2.2). Additionally, I consider factors related to the objective function for training $\mathcal{M}$ including (i) *curvature smoothness* (O1), (ii) *distinguishability of model observables (between data records, demographic groups, datasets, and models)* (O2), and (iii) *distance of data records in $\mathcal{D}_{tr}$ to the decision boundary* (O3). Finally, I consider *capacity* (the same as M1) as the model-related factor. I elaborate them below.

Memorization Factors Related to Dataset Characteristics

D2 **Tail length of $\mathcal{D}_{tr}$'s distribution:** Consider the distribution of the number of data records in $\mathcal{D}_{tr}$ across different classes (for classifier). This distribution can vary by either being a uniform distribution or a long-tailed distribution. A uniform distribution indicates that the number of data records in $\mathcal{D}_{tr}$ is balanced across different classes. For a *long-tailed* distribution, the majority of the data records belong to the classes from the "head" of the distribution, while data from tail classes is scarce [226, 227]. Data records from the tail classes are either rare or outliers, and thus, $\mathcal{M}$ is more likely to memorize them [77].

D3 **Number of input attributes:** As the number of input attributes increases, a sufficiently large $\mathcal{M}$ can memorize more about the individual attributes and their correlations [228]. However, for a fixed capacity, increasing the number of attributes can force $\mathcal{M}$ to share capacity across attributes, reducing the memorization of individual attributes [73].
D4 **Priority of learning stable attributes:** Recall the notions of stable and spurious attributes from Chapter 2: Section 2.2. When $\mathcal{M}$ prioritizes learning stable attributes over spurious attributes, it generalizes better, making it invariant to changes in other attributes, dataset, or distribution [90, 91, 229]. On the contrary, prioritizing spurious attributes increases $\mathcal{M}$'s memorization. Measuring stable attributes is challenging as there are no ground truths. Prior works have used *mean rank metric* to measure the distance between any two inputs over their learned representation by $\mathcal{M}$ [230, 91]. Its value is small if the learned representations of some attributes between two inputs are the same (i.e., corresponds to stable attributes). Hence, a low mean rank suggests that $\mathcal{M}$ prioritized learning stable attributes.

Memorization Factors Related to Objective Function

O1 **Curvature smoothness:** The convergence of the loss during training on different data records in $\mathcal{D}_{tr}$ towards a minima depends on the smoothness of the objective function. Since different data records in $\mathcal{D}_{tr}$ have varying levels of difficulty in terms of being learned by $\mathcal{M}$, a less smooth curvature can lead to the convergence to a local minima around the difficult data records [231, 76, 232]. Thus, the data records in $\mathcal{D}_{tr}$ exhibit varying memorization patterns, evident by the difference in the influence of data records on the model observables (e.g., [233]).
O2 **Distinguishability of model observables:** Due to the varying difficulty in learning different data records in $\mathcal{D}_{tr}$, and thus variable memorization, $\mathcal{M}$ exhibits distinguishability in model observables. I identify three subtypes based on distinguishability in model observables between O2.1 datasets (e.g., between data records inside and outside $\mathcal{D}_{tr}$), O2.2 demographic groups (e.g., between males and females), O2.3 models (e.g., between models trained on datasets with different distributions [12]).
O3 **Distance of $\mathcal{D}_{tr}$ data records to decision boundary:** Memorized data records are closer to the decision boundary [231] as also seen in Chapter 2: Section 2.2. This is because removing data records that are closer to the decision boundary is likely to change the boundary upon retraining. This results in changes to the predictions for the removed data records; that is, these records are not leave-one-out stable and are therefore memorized.

Memorization Factors Related to Model Characteristics

For model characteristics, I consider the *capacity of* $\mathcal{M}$ which is the same as M1. Increasing $\mathcal{M}$'s capacity generally increases memorization by fitting a more complex decision boundary [234, 58, 86, 87]. When $\mathcal{M}$ is too simple to effectively fit a given $\mathcal{D}_{tr}$, increasing $\mathcal{M}$'s capacity typically leads to better generalization and lower overfitting. However, continuing to increase $\mathcal{M}$'s capacity after it already generalizes well can increase memorization of data records.

I summarize the various factors in Table 3.1, and discuss the completeness of the underlying causes and factors, as well as extensions of the framework in Section 3.4.

Table 3.1: **Summary of factors in the framework:** I include factors that influence overfitting and memorization, covering characteristics of data, model, and objective function.

Factor	Description
D1	Size of $\mathcal{D}_{tr}$
D2	Tail length of $\mathcal{D}_{tr}$'s distribution
D3	Number of input attributes
D4	Priority of learning stable attributes
O1	Curvature smoothness of the objective function
O2 (O2.1)	Distinguishability of observables between datasets
O2 (O2.2)	Distinguishability of observables across subgroups
O2 (O2.3)	Distinguishability of observables across models
O3	Distance of $\mathcal{D}_{tr}$ data records to the decision boundary
M1	$\mathcal{M}$'s capacity

3.2 Understanding Unintended Interactions

I relate the factors driving overfitting and memorization to defenses (Section 3.2.1) and risks (Section 3.2.2). Table 3.2 summarizes these relationships. I then review unintended interactions identified in prior work and situate them within the framework (Section 3.2.3), and conclude by presenting the guideline to anticipate such interactions (Section 3.2.4).

3.2.1 Revisiting Defenses for ML

I first explore how the effectiveness of defenses correlates with changes in various factors. Among all the variants of different defenses in Chapter 2: Section 2.4, I focus on the specific defenses explored in the literature on unintended interactions. This includes:

- adversarial training (`EvsnRob-In` indicated as `EvsnRob` for brevity);
- outlier removal-based poisoning robustness (`PoisnRob-Pre` indicated as `PoisnRob` for brevity);
- backdoor-based model watermarking (`MdlWM-Pre` indicated as `MdlWM` for brevity);
- post-hoc model fingerprinting (`Fngrprnt-Post` indicated as `Fngrprnt` for brevity);
- differential privacy using DPSGD (`DiffPriv-In` indicated as `DiffPriv` for brevity);
- in-training group fairness (`GpFair-In` indicated as `GpFair` for brevity); and
- post-hoc model explanations (`Expl-Post` indicated as `Expl` for brevity).

`EvsnRob` (Adversarial Training)

$\mathcal{M}$ learns spurious attributes that are essential for achieving high test accuracy ϕ_{acc}. However, this makes $\mathcal{M}$ susceptible to evasion through adversarial examples [235]. $\mathcal{M}^{evsnrob}$[1] is forced to learn stable attributes [138, 235], which improves robustness. Additionally, `EvsnRob` includes a regularization term that minimizes the maximum loss from adversarial examples, resulting in smoother curvature [99]. This creates an explicit trade-off between generalization and robustness [236], as evidenced by higher overfitting and a drop in ϕ_{acc} ($\vee$) [235]. Furthermore, increasing the size of $\mathcal{D}_{tr}$ can reduce overfitting [138]. On the other hand, increasing the tail length of $\mathcal{D}_{tr}$'s distribution lowers robustness for the tail classes [237, 61]. Training with `EvsnRob` is typically accompanied by an increase in the distance of data records in $\mathcal{D}_{tr}$ from the decision boundary [238]. Prior work has also shown that $\mathcal{M}^{evsnrob}$ exhibits greater distinguishability in $\mathcal{M}$'s predictions for data records both in and out of $\mathcal{D}_{tr}$ [58]. Finally, $\mathcal{M}^{evsnrob}$ should have high capacity to effectively train on adversarial examples [99].

`PoisnRob` (Poison Robustness)

Poisoned data records are typically outliers relative to $\mathcal{D}_{tr}$'s distribution. Hence, `PoisnRob` typically involves identifying such outliers by measuring the influence of data records on model observables [145] and removing them. A longer tail in $\mathcal{D}_{tr}$ increases the number of memorized outliers making their detection easy [239]. Some of these outliers contribute to $\mathcal{M}$'s

[1]Recall from Chapter 2, we use $\mathcal{M}^{evsnrob}$ for model trained with defense against `Evsn`. We use similar notations for models trained with other defenses taken from Chapter 2.

accuracy [77, 85, 74]. Thus, removing them increases overfitting in $\mathcal{M}^{psnrob}$ and a drop in ϕ_{acc} ($\vee$) [145, 233, 240, 146].

`MdlWM` (Model Watermarking)

Ideally, $\mathcal{M}^{mdlwm}$ should have similar ϕ_{acc} to $\mathcal{M}$, but in practice, $\mathcal{M}^{mdlwm}$ typically incurs a drop in ϕ_{acc} ($\vee$) [241, 169, 242]. Since watermarks are essentially outliers, a long tail in $\mathcal{D}_{tr}$ enables stealthier watermarks typically from tail classes [243]. Additionally, having a lower distinguishability in model observables for watermarks between $\mathcal{M}$ and $\mathcal{M}^{sur}$, while remaining distinct from independently trained models, helps identify $\mathcal{M}^{sur}$ [241]. Finally, a high capacity $\mathcal{M}$ aids memorizing the watermarks, thus, improving its effectiveness [241].

`Fngrprnt` (Fingerprinting)

There is no explicit impact on overfitting, memorization, or ϕ_{acc} as `Fngrprnt` does not require retraining ($\sim$). A lower distinguishability in fingerprints between $\mathcal{M}$ and $\mathcal{M}^{sur}$, but distinct from independently trained models, is preferred for effectively identifying $\mathcal{M}^{sur}$ [244, 172]. The effectiveness of fingerprints relies on $\mathcal{M}$'s overfitting and memorization. For instance, dataset inference hinges on the distinguishability of model observables between data records inside and outside $\mathcal{D}_{tr}$, for both $\mathcal{M}$ and $\mathcal{M}^{sur}$ [174]. This helps identify if they share overlapping $\mathcal{D}_{tr}$. Similarly, fingerprints based on adversarial examples rely on the effectiveness of transferring adversarial examples to $\mathcal{M}^{sur}$ [172, 174, 170, 171, 245]. For effective transfer of adversarial examples, a smaller distance between data records in $\mathcal{D}_{tr}$ and the decision boundary is preferred, as it facilitates the generation of effective adversarial examples [172, 171, 170].

`DiffPriv` (Differential Privacy)

`DiffPriv` reduces the memorization of data records in $\mathcal{D}_{tr}$ by limiting their influence on model observables. Adding noise to gradients in `DiffPriv` regularizes $\mathcal{M}^{dp}$ which curbs overfitting but with a drop in ϕ_{acc} ($\vee$) [177, 246, 247]. This reduces distinguishability in model observables for data records inside and outside $\mathcal{D}_{tr}$ [177], and decreases the distance of $\mathcal{D}_{tr}$ data records to the boundary [248]. Also, learning stable attributes typically results in better privacy guarantees as evidenced in causal models [90]. Prior works have speculated that `DiffPriv` reduces the curvature smoothness of the objective function [248, 249]. It has also been noted that having a long-tailed $\mathcal{D}_{tr}$ distribution provides lower privacy for tail classes and a disproportionate impact on minority demographic groups [250, 251]. Finally, having $\mathcal{M}$ with low capacity is preferred to reduce the noise required for privacy, and thus, reduce the impact on ϕ_{acc} [252].

`GpFair` (Group Fairness)

For fairness via in-processing algorithms, the regularization term explicitly creates a trade-off between fairness and ϕ_{acc} [253, 254], exacerbating overfitting with a drop in $\mathcal{M}^{gpfr}$'s accuracy ($\vee$) [213, 255, 253, 254]. By definition, training for fairness reduces distinguishability in model observables across demographic groups [256, 211]. Further, $\mathcal{M}^{gpfr}$ prioritizes learning stable attributes that do not overlap with sensitive attributes [257, 256]. In order to ensure equitable behaviour, $\mathcal{M}^{gpfr}$ typically memorizes minority data records, thereby decreasing their distance to the decision boundary [258, 259].

`Expl` (Explanations)

There is no explicit impact on overfitting, memorization, or ϕ_{acc} ($\sim$) as `Expl` does not require modifying $\mathcal{M}$. However, the quality of `Expl` relies on various factors related to overfitting and memorization. Increasing the number of input attributes, reduces the influence of individual attributes on $\mathcal{M}$ which affects the quality of `Expl` [260]. Also, `Expl` exhibits distinguishability across demographic groups [261]. The quality of `Expl` is tied to $\mathcal{M}$'s overfitting: it exhibits better approximation for data records in $\mathcal{D}_{tr}$ than for those in $\mathcal{D}_{te}$ [260]. Therefore, `Expl` exhibits distinguishability between data records inside and outside $\mathcal{D}_{tr}$ [260].

I summarize the relation between effectiveness of defenses with various factors in the left column of Table 3.2. For a defense **`Def`** and a factor `<f>`, $\uparrow$ indicates that **`Def`** positively correlates with `<f>`, while $\downarrow$ indicates a negative correlation.

3.2.2 Revisiting Risks in ML

I will now discuss how changes in various factors correlate with changes in susceptibility to different risks. I focus on all risks in Chapter 2: Section 2.3, except unauthorized data usage (`UnauthData`), where unintended interactions of its corresponding defense, i.e., data watermarking (`DtWM`), will be analogous to model watermarking (`MdlWM`).

`Evsn` (Evasion)

Having a long-tailed $\mathcal{D}_{tr}$ increases the susceptibility to `Evsn`, as the adversarial examples from tail classes are more effective [227, 226]. Additionally, reducing the smoothness of the objective function's curvature makes it easier to generate adversarial examples [99]. It has also been

Table 3.2: **Relation between factors with defenses/risks:** Correlation between the effectiveness of a defense `Def` and susceptibility to a risk `Rk` with a factor <f>: ↑ indicates `Def` or `Rk` positively correlates with <f>; ↓ indicates a negative correlation.

Defenses (<↑ or ↓>, <f>)	**Risks (<f>, <↑ or ↓>)**
`EvsnRob` (Adversarial Training): • D1 ↑, $\|\mathcal{D}_{tr}\|$ [138] • D2 ↓, tail length [237, 61] • D4 ↑, priority for learning stable attributes [138] • O1 ↑, curvature smoothness [99] • O2.1 ↑, distinguishability across datasets [262] • O3 ↑, distance to decision boundary [238] • M1 ↑, model capacity [99] **`PoisnRob` (Outlier Removal):** • D2 ↑, tail length [239] **`MdlWM` (Watermarking):** • D2 ↑, tail length [243] • O2.3 ↓, distinguishability in $\mathcal{M}^{mdlwm}$ and $\mathcal{M}^{sur}$ [241] • M1 ↑, model capacity [241] **`Fngrprnt` (Fingerprinting):** • O2.3 ↓, distinguishability between $\mathcal{M}$ and $\mathcal{M}^{sur}$ [244, 172] • O3 ↓, distance to decision boundary [172, 171, 170] **`DiffPriv` (Differential privacy):** • D2 ↓, tail length [251, 250] • D4 ↑, priority for learning stable attributes [90] • O1 ↓, curvature smoothness [248, 249] • O2.1 ↓, distinguishability across datasets [177] • O3 ↓, distance to decision boundary [248] • M1 ↓, model capacity [252] **`GpFair` (Group Fairness):** • D4 ↑, priority for learning stable attributes [263, 257, 256] • O2.2 ↓, distinguishability across subgroups [256, 211] • O3 ↓, distance to decision boundary [259] **`Expl` (Explanations):** • D3 ↓, number of input attributes [260] • O2.1 ↑, distinguishability across datasets [260] • O2.2 ↑, distinguishability across subgroups [261]	**`Evsn` (Evasion):** • D2 ↑, tail length [227, 226] • O1 ↓, curvature smoothness [99] • O3 ↓, distance to decision boundary [248] **`Poisn` (Poisoning):** • D2 ↑, tail length [149, 264, 243] • M1 ↑, model capacity [241] **`ModExt` (Unauthorized Model Ownership):** • M1 ↓, model capacity [106, 107] **`MemInf` (Membership Inference):** • D1 ↓, $\|\mathcal{D}_{tr}\|$ [265, 266] • D2 ↑, tail length [76, 267] • D4 ↓, priority for learning stable attributes [91, 90] • O2.1 ↑, distinguishability across datasets [266] • O3 ↓, distance to decision boundary [260] • M1 ↑, model capacity [262] **`DtRecon` (Data Reconstruction):** • D2 ↑, tail length [268] • D3 ↓, number of input attributes [269] • O2.1 ↑, distinguishability across datasets [270, 271] • O2.2 ↑, distinguishability across subgroups [272] **`AttInf` (Attribute Inference):** • D2 ↑, tail length [115] • D4 ↓, priority for learning stable attributes [91, 273] • O2.2 ↑, distinguishability across subgroups [256] **`DistInf` (Distribution Inference):** • D2 ↑, tail length [274, 275] • D4 ↑, priority for learning stable attributes [121] • O2.1 ↑, distinguishability across datasets [121, 12] • O2.2 ↑, distinguishability across subgroups [256] • M1 ↑, model capacity [274] **`Discr` (Discriminatory behaviour):** • D2 ↑, tail length [55, 124] • O2.2 ↑, distinguishability across subgroups [211]

noted that data records close to $\mathcal{M}$'s decision boundary, which are likely to be memorized, require smaller perturbations to induce misclassification [276, 277]. Hence, the risk of `Evsn` decreases when the decision boundary is pushed away from the data records in $\mathcal{D}_{tr}$ [248]. When overfitting and memorization are pronounced, the decision boundary becomes more

complex and more closely fits data records in $\mathcal{D}_{tr}$, making it easier to find adversarial examples. Hence, susceptibility to Evsn is higher [277].

Poisn (Poisoning)

A longer-tailed $\mathcal{D}_{tr}$ increases the effectiveness of Poisn, as poisons from tail classes are more subtle [149, 264, 243]. Following watermarking literature that uses poisons (specifically backdoors), the effectiveness of poisons is higher for $\mathcal{M}$ with larger capacity, due to increased memorization [241].

ModExt (Unauthorized Model Ownership)

ModExt is less successful if $\mathcal{M}$ overfits or memorizes: ϕ_{acc} of $\mathcal{M}$ is lower and consequently, labels from $\mathcal{M}$ for training $\mathcal{M}^{sur}$ are of poor quality. In other words, the poor generalization of $\mathcal{M}$ worsens the generalization of $\mathcal{M}^{sur}$. Hence, $\mathcal{M}^{sur}$ is likely to have low ϕ_{acc} [278]. Also, for a given $\mathcal{M}^{sur}$, increasing $\mathcal{M}$'s capacity decreases susceptibility to ModExt, due to the higher quality of predictions used to train $\mathcal{M}^{sur}$ [107, 106].

MemInf (Membership Inference)

Overfitting is correlated with an increase in susceptibility to MemInf. Hence, the risk of MemInf decreases with larger $|\mathcal{D}_{tr}|$, which improves the generalization of $\mathcal{M}$ [265, 266]. Further, when $\mathcal{M}$ prioritizes learning stable attributes, susceptibility to MemInf decreases due to better generalization [91, 90]. Also, higher distinguishability between data records inside and outside $\mathcal{D}_{tr}$ increases susceptibility to MemInf [266, 265, 51]. It has also been noted that memorized data records exhibit higher influence on model observables, rendering them more susceptible to MemInf [240, 267]. Hence, a longer-tailed distribution increases the fraction of memorized outliers, thereby increasing their susceptibility to MemInf [76, 267]. Data records located farther from the decision boundary have lower memorization and are less susceptible to MemInf [260]. Finally, increasing $\mathcal{M}$'s capacity amplifies memorization of $\mathcal{D}_{tr}$, which increases MemInf [58].

DtRecon (Data Reconstruction)

Higher overfitting and memorization of data records in $\mathcal{D}_{tr}$ increase their influence on model observables, leading to more effective reconstruction of the data records [271, 279, 280, 281, 268].

Also, higher distinguishability in model observables between data records inside and outside $\mathcal{D}_{tr}$, increases susceptibility to DtRecon [270]. Additionally, outliers, typically belonging to the tail of $\mathcal{D}_{tr}$ distribution, are more effectively reconstructed [268]. Further, increasing the number of input attributes reduces the accuracy of input reconstruction [269]. Finally, higher distinguishability in model observables across different demographic groups also increases the susceptibility to DtRecon [272].

AttInf (Attribute Inference)

Prior work has empirically found that overfitting has no discernible impact on the susceptibility to AttInf [278]. However, memorization of sensitive attributes increases the susceptibility to AttInf [232]. Hence, a long-tailed $\mathcal{D}_{tr}$ increases the fraction of memorized data records, increasing their susceptibility to AttInf [115]. Further, prioritizing the learning of stable attributes decreases the susceptibility to AttInf [91, 273]while a higher distinguishability across demographic groups increases AttInf [256].

DistInf (Distribution Inference)

Prior works have empirically found that overfitting reduces the susceptibility to DistInf, although the reason remains unexplained [121, 274]. Further, an increase in the tail length positively correlates with the susceptibility to DistInf [274, 275]. Prioritizing the learning of stable attributes amplifies the risk of DistInf, especially when the stable attributes being memorized correspond to the distributional property being inferred [121]. Higher distinguishability in model observables across datasets with different distributional properties increases the risk of DistInf. [12, 121]. Extending the observation from AttInf [256], I speculate that the effectiveness of DistInf depends on distinguishability of model observables across demographic groups. A higher distinguishability is likely to result in a higher susceptibility to DistInf. Finally, increasing $\mathcal{M}$'s capacity typically increases DistInf [274].

Discr (Discriminatory Behaviour)

Overfitting makes bias more prominent, as evidenced by the disparity in ϕ_{acc} between minority and majority demographic groups [282]. Increased memorization of minority demographic groups also contributes to an increase in Discr [258]. Also, a long-tailed $\mathcal{D}_{tr}$, which increases the fraction of memorized data records, worsens Discr [55, 124]. Finally, increased

distinguishability in model observables across different demographic groups correlates with increased susceptibility to `Discr` [211, 256].

The above discussion is summarized in the right column of Table 3.2. For a risk **`Rk`** and a factor `<f>`, ↑ indicates **`Rk`** positively correlates with `<f>`, and ↓ indicates a negative correlation.

3.2.3 Surveying Unintended Interactions

I now present a survey of various unintended interactions explored in the literature and situate them within this framework. I selected papers using Google Scholar, focusing on those published in top-tier venues. Next, I examined their citations and related work to identify additional studies. I also used existing surveys (e.g., Gittens et al. [219]) to ensure that the coverage of prior work is comprehensive.

I present the unintended interactions with security defenses in Table 3.3, and with remaining defenses in Table 3.4. I mark the type of interaction (indicated as "**I**") between **`Def`** and **`Rk`** as 🔴 if "an increase (decrease) in effectiveness of **`Def`** correlates with an increase (decrease) in **`Rk`**" and 🟢 if "an increase (decrease) in effectiveness of **`Def`** correlates with a decrease (increase) in **`Rk`**". When **I** is ⚪, prior work has not studied this interaction, and the middle columns are left empty. When **I** is 🔴 or 🟢, empty middle columns indicate that prior work merely identifies the type of interaction without evaluating the impact of the factors.

Additionally, I use the factors from Table 3.1 to provide a more nuanced understanding of the interactions. I mark a factor with ● if there are empirical results regarding its influence, ⊛ if there are theoretical results, and ○ if it is only conjectured. Exceptions to any interactions are listed under "References" and discussed in the text below.

`EvsnRob` (Adversarial Training)

🟢 **`Evsn` (Evasion)** is less effective with `EvsnRob` which is specifically designed to resist adversarial examples [99, 135, 53]. Using the min-max objective function increases the smoothness of the curvature and pushes the decision boundary away from data records [135], thereby reducing the likelihood of generating adversarial examples [99]. Also, $\mathcal{M}^{evsnrob}$ requires a high capacity to learn a complex decision boundary to fit adversarial examples [99]. Increasing the tail length of $\mathcal{D}_{tr}$ in turn increases the effectiveness of adversarial examples from the tail classes despite using `EvsnRob` [226, 227].

🔴 **`Poisn` (Poisoning)** can be more effective with `EvsnRob` [283]. Recall that $\mathcal{M}^{evsnrob}$ is forced to learn stable attributes. Hence, it might seem that poisons, generated by manipulating spu-

Table 3.3: **Unintended interactions with security defenses**: *Type of interaction (I)*: ●→ risk increases; ●→ risk decreases; ●→ unexplored. I indicate exceptions to these under "References" and discuss them in the text. *Underlying factors:* ● → empirical; ⊛ → theoretical and ○ → conjectured. Under O2, I write O2.1, O2.2, and O2.3 as 1, 2, and 3 respectively. I mark the factors evaluated for conjectured interactions as ∗ (See Section 3.3).

Defenses	Risks	I	OVFT		Memorization					Both	References
			D1	D2	D3	D4	O1	O2	O3	M1	
EvsnRob	Evsn	●		●			●		●	●	[135, 99, 226, 227]
	Poisn	●									[283, 284]
	ModExt	●	○								[62] ([278]: ●)
	MemInf	●	⊛, ●					1: ●		●	[262, 59]
	DtRecon	●				○				●	[285, 286]
	AttInf	●									
	DistInf	●				○					[121]
	Discr	●		⊛, ●							[61, 287, 237, 60]
PoisnRob	Evsn	●									[144]
	Poisn	●									[56]
	ModExt	●									
	MemInf	●		●							[76, 240]
	DtRecon	●									
	AttInf	●		●							[115]
	DistInf	●									
	Discr	●	●	○							[288]
MdlWM	Evsn	●									
	Poisn	●		○							[289, 241, 290, 291]
	ModExt	●		○				3: ●	●		[169, 241, 242]
	MemInf	●		○				1: ●	●		[232, 292]
	DtRecon	●		○				1: ●	●		[232]
	AttInf	●		○				2: ●	●		[232]
	DistInf	●	⊛, ●	○				1: ●	●	●	[274, 275]
	Discr	●		○							[243]
Fngrprnt	Evsn	●						3: ●	●		[172, 170, 171]
	Poisn	●									
	ModExt	●		●				3: ●	●		[173, 244, 174, 172]
	MemInf	●							●		[174]
	DtRecon	●									
	AttInf	●									
	DistInf	●									
	Discr	●									

rious attributes, do not affect $\mathcal{M}^{evsnrob}$'s predictions [284]. However, Wen et al. [283] design poisons to degrade $\mathcal{M}^{evsnrob}$'s accuracy by manipulating the stable attributes.

● **ModExt (Unauthorized model ownership)** is easier for $\mathcal{M}^{evsnrob}$ compared to $\mathcal{M}$ [62]. However, no reasons were explored. I conjecture that since $\mathcal{M}^{evsnrob}$ has uniform predictions on neighboring data records [108, 293, 99, 53], $\mathcal{M}^{sur}$ requires fewer queries compared to $\mathcal{M}$. The above interaction assumes $\mathcal{A}dv$ is a malicious suspect. For a malicious accuser, EvsnRob

can mitigate false claims [92], which decreases ModExt (●). This is an example where the nature of interaction varies depending on the threat model.

● **MemInf (Membership inference)** is more effective with EvsnRob [58, 59]. Recall that EvsnRob modifies $\mathcal{M}^{evsnrob}$'s decision boundary to memorize adversarial examples [58]. This increases the influence of some records on model observables. Consequently, the distinguishability between data records inside and outside $\mathcal{D}_{tr}$ is higher [58]. Hence, the susceptibility to MemInf increases. Hayes et al. [59] provably show that increasing $|\mathcal{D}_{tr}|$ lowers overfitting in $\mathcal{M}^{evsnrob}$ and hence, reduces the risk of MemInf. Further, MemInf increases with $\mathcal{M}^{evsnrob}$'s capacity due to higher memorization.

● **DtRecon (Data reconstruction)** is more effective with EvsnRob [285, 286]: $\mathcal{M}^{evsnrob}$ has more interpretable gradients than standard $\mathcal{M}$, which increases the quality of reconstruction of data records [294]. Further, stable attributes across inputs (e.g., same background across multiple images) are easier to reconstruct.Finally, it has been shown that lower capacity of $\mathcal{M}^{evsnrob}$ results in higher susceptibility to DtRecon [285]. The authors speculate that this is because smaller models cannot learn adversarial examples as effectively as larger models (i.e., the effectiveness of EvsnRob is reduced).

● **DistInf (Distribution inference)** is easier as $\mathcal{M}^{evsnrob}$ tends to learn stable attributes that match distributional properties being inferred [121]. However, increasing ϵ_{rob} reduces the susceptibility to DistInf as $\mathcal{M}^{evsnrob}$'s ϕ_{acc} decreases.

● **Discr (Discriminatory behaviour)** increases with EvsnRob as evident from the disparity in ϕ_{acc} of $\mathcal{M}^{evsnrob}$ across different classes [237, 61, 287]. Longer tail contributes to the tail classes having lower robustness [237, 61]. Also, increasing ϵ_{rob}, increases the class-wise disparity [60]. Prior works have empirically shown that size of $\mathcal{D}_{tr}$, $\mathcal{M}^{evsnrob}$'s capacity, type of $\mathcal{D}_{tr}$ (e.g., images, tabular), and $\mathcal{M}^{evsnrob}$'s architecture and optimization algorithms, *do not* influence the interaction [237, 61].

PoisnRob (Poison Robustness)

● **Evsn (Evasion)** is less effective with PoisnRob [144]. Adversarial examples are outliers compared to $\mathcal{D}_{tr}$'s distribution that can be detected before passing them as inputs [144].

● **Poisn (Poisoning)** is less effective with PoisnRob which is specifically designed to detect and remove poisons as outliers during pre-training [56, 149]. During training, the influence of outliers (as poisons) on model observables can be minimized [264].

● **MemInf (Membership inference)** is more effective with PoisnRob (i.e., removing poisoning followed by retraining) [76]. Memorized outliers are more susceptible to MemInf but after

these outliers are removed, $\mathcal{M}^{psnrob}$ tends to memorize other data records, making them more susceptible than before [76]. Furthermore, prior work has shown that varying the size of $\mathcal{D}_{tr}$, capacity of $\mathcal{M}^{psnrob}$, or the number of duplicates in $\mathcal{D}_{tr}$, does *not* have a signficant impact on the suscpetibility to `MemInf` [76].

● **`AttInf` (Attribute inference)** is more effective as retraining after removal of memorized outliers constituting the long-tail of $\mathcal{D}_{tr}$, makes new data records susceptible [115]. The reasoning is the same as the interaction with `MemInf` above.

● **`Discr` (Discriminatory behaviour)** increases as outliers constitute a statistical minority. Here, `PoisnRob` increases bias against minorities by excessively identifying them as outliers [295]. As $|\mathcal{D}_{tr}|$ increases, more minority data records are classified as outliers, exacerbating the bias [295].

`MdlWM` (Watermarking)

● **`Poisn` (Poisoning)**, specifically using backdoors, are used by watermarking schemes requiring $\mathcal{M}$ to be susceptible to it [289, 241, 165, 291]. These schemes inject data records similar to outliers to increase the likelihood of their memorization by $\mathcal{M}$. Hence, long-tail distribution of $\mathcal{D}_{tr}$ can help generate effective and discreet backdoors as watermarks.

● **`ModExt` (Unauthorized model ownership)** can be detected with `MdlWM` [169, 241, 242]. Sufficient capacity in $\mathcal{M}^{mdlwm}$ is necessary to memorize watermarks; therefore, increasing model capacity can enhance the effectiveness of watermarking [242]. Also, longer tails help with generating more discreet watermarks.

● **`MemInf` (Membership inference)**, **`DtRecon` (Data reconstruction)**, **`AttInf` (Attribute inference)**, **`DistInf` (Distribution inference)** do not have any prior evaluations with watermarks. However, prior work has explored stealthy poisoning where mislabeled data records are added to $\mathcal{D}_{tr}$ to increase different privacy risks without a drop in ϕ_{acc} [232, 292, 274, 275]. I liken this form of poisoning to `MdlWM` [289, 241, 165, 291], and use their observations to justify different interactions. I conjecture that adding watermarks from tail classes, similar to poisoning, will increase the privacy risks. Hence, a longer tail is likely to increase different privacy risks. Further, memorization of watermarks increases their influence on model observables. This increases the distinguishability in model observables across datasets, demographic groups, and models. Moreover, the decision boundary is more complex from memorizing the watermarks, consequently increasing the susceptibility of outliers closer to the decision boundary [232, 292]. Additionally, in case of `DistInf`, prior works have shown that the risk increases with lower capacity, and larger $\mathcal{D}_{tr}$ size [274, 275].

● **`Discr` (Discriminatory behaviour)** could worsen `Discr`, especially if it relies on data records from a majority demographic group, which can increase bias in $\mathcal{D}_{tr}$ [243]. Additionally, watermarks (with incorrect labels) belonging to tail classes are more effectively memorized for ownership verification [243, 296, 297]. Thus, increasing the tail length is likely to improve the effectiveness of watermarks.

`Fngrprnt` (Fingerprinting)

● **`Evsn` (Evasion)** can be useful for fingerprinting where the fingerprints are essentially adversarial examples, necessitating $\mathcal{M}$'s susceptibility to `Evsn` [172, 171, 170]. Hence, when $\mathcal{M}$ is robust against evasion, fingerprints are less effective [172]. Such fingerprints require $\mathcal{D}_{tr}$ records to be closer to decision boundary to allow effectively finding adversarial examples [170]. Here, overfitting and memorization make the decision boundary complex, which helps generate fingerprints. To transfer these fingerprints from $\mathcal{M}$ to $\mathcal{M}^{sur}$, the distinguishability between the model observables should be low but distinct from independent models [172, 171, 170].

● **`ModExt` (Unauthorized model ownership)** can be checked using fingerprints. Unique intermediate activations for some inputs can be used as a fingerprint which have low distinguishability between $\mathcal{M}$ and $\mathcal{M}^{sur}$ but distinct from independent models [244]. Fingerprints based on adversarial examples can be generated more effectively with a longer-tailed $\mathcal{D}_{tr}$ distribution and a smaller distance between data records in $\mathcal{D}_{tr}$ and the decision boundary, resulting in more successful detection of `ModExt` [172, 171, 170, 174].

● **`MemInf` (Membership inference)** can be detected by identifying common data records between $\mathcal{M}$ and $\mathcal{M}^{sur}$, which requires $\mathcal{M}$ to be susceptible to such analysis. Dataset inference uses a weak membership inference attack to identify data records close to $\mathcal{M}$'s decision boundary that constitute its fingerprint [174]. Specifically, it distinguishes between the distribution of distances of data records in $\mathcal{D}_{tr}$ and any other independent dataset to the decision boundary, and *not* any individual data records.

`DiffPriv` (Differential Privacy)

● **`Evsn` (Evasion)** is more effective against $\mathcal{M}^{dp}$ compared to a non-private $\mathcal{M}$. DPSGD reduces the distance between $\mathcal{D}_{tr}$ records and the decision boundary, making it easier to generate adversarial examples [248]. Additionally, training with DPSGD reduces curvature smoothness, which can facilitate the generation of effective adversarial examples and increase `Evsn` [248, 330, 249, 331]. $\mathcal{M}^{dp}$ can create a false sense of robustness via gradient masking,

Table 3.4: **Unintended interactions with defenses for privacy, fairness, and transparency**: Symbol and their descriptions are the same as in Table 3.3.

Defenses	Risks	I	OVFT		Memorization					Both	References
			D1	D2	D3	D4	O1	O2	O3	M1	
DiffPriv	Evsn	🔴					●		●		[248, 249] ([298]:🟢)
	Poisn	🔴									[299, 300, 301]
	ModExt	⚪									
	MemInf	🟢				⊛, ●		1: ●			[246, 247, 278, 302]
	DtRecon	🟢						1: ●			[303, 304]
	AttInf	🟢						2: ○			[278]
	DistInf	🟢						●			[305, 121] ([121]: 🔴)
	Discr	🔴		○					●		[251, 306, 231, 307, 308]
GpFair	Evsn	🔴					●		⊛, ●		[259]
	Poisn	🔴									[309, 296, 310, 311]
	ModExt	⚪									
	MemInf	🔴	●						●	●	[258, 312]
	DtRecon	⚪			✳			2: ✳			Our conjecture: 🟢
	AttInf	🟢						2:⊛,●			[256] ([313]: 🔴)
	DistInf	⚪									Our Conjecture: 🟢
	Discr	🟢	●					2:●			[314, 211, 315, 316]
Expl	Evsn	🔴									[317, 318]
	Poisn	🔴			●		⊛, ●			●	[319, 320, 321, 322]
	ModExt	🔴									[323, 324, 318]
	MemInf	🔴	●		●			1: ●	●	●	[260, 325, 318, 326]
	DtRecon	🔴		⊛, ●	●						[260, 327]
	AttInf	🔴						2: ○			[261]
	DistInf	⚪			✳			1: ✳		✳	Our Conjecture: 🔴
	Discr	🔴		●	●					●	[328, 329] ([209, 208]: 🟢)

i.e., modifying gradients so that they cannot be used to optimize adversarial examples. However, it has been shown that $\mathcal{M}^{dp}$ remains susceptible to adversarial examples when these are optimized without using its gradients.

On the other hand, some prior work shows that training with DPSGD can lead to adversarial robustness, depending on the choice of hyperparameters (🟢) [248, 298]. Further investigation is required to understand the conditions under which DPSGD and adversarial robustness align.

🔴 **`Poisn` (Poisoning)** can be more effective against $\mathcal{M}^{dp}$. Prior work suggests that, since `DiffPriv` reduces the influence of individual data points on model observables and constrains gradient magnitudes, it should improve robustness against `Poisn` [301, 300]. However, it has been shown that the choice of hyperparameters (e.g., noise multiplier and gradient clipping) is the primary reason for the observed improvement in robustness [299]. Further investigation is required to understand the root cause of this interaction.

● **MemInf (Membership inference)** is less effective on $\mathcal{M}^{dp}$ due to DiffPriv's regularizing effect, as evidenced by reduced overfitting [177]. This reduces the distinguishability between data records inside and outside $\mathcal{D}_{tr}$, thereby decreasing susceptibility to MemInf [177, 246, 278]. Additionally, training models to prioritize causal (stable) attributes provides stronger privacy guarantees while further reducing the risk of MemInf [90]. Finally, DiffPriv provably bounds the risk of MemInf for i.i.d. data [246, 247]. However, this bound does not hold for MemInf on non-i.i.d. data [302]. Moreover, duplicates in $\mathcal{D}_{tr}$ weaken the bound, thus, increasing MemInf.

● **DtRecon (Data reconstruction)** is less effective against $\mathcal{M}^{dp}$, as DiffPriv reduces the distinguishability between data records inside and outside $\mathcal{D}_{tr}$ [303]. The reduced influence of individual data records on model observables further decreases susceptibility to DtRecon.

● **AttInf (Attribute inference)** is less effective against $\mathcal{M}^{dp}$ [278], but prior work has not explored the underlying reasons. I conjecture that DiffPriv, by reducing the memorization of sensitive attributes and their influence on model observables, decreases distinguishability across demographic groups [145]. This, in turn, reduces susceptibility to AttInf.

● **DistInf (Distribution inference)** is less effective, assuming $\mathcal{A}dv$ does not know $\mathcal{M}^{dp}$'s training hyperparameters and therefore cannot adapt the attack [121]. DiffPriv has a regularizing effect that reduces $\mathcal{M}^{dp}$'s ability to learn the distribution of $\mathcal{D}_{tr}$. This, in turn, reduces the distinguishability of model observables across datasets with different distributional properties, thereby decreasing susceptibility to DistInf [121]. However, if $\mathcal{A}dv$ knows the training hyperparameters of $\mathcal{M}^{dp}$, they can adapt their attack to increase DistInf (●) [121].

● **Discr (Discriminatory behaviour)** increases for $\mathcal{M}^{dp}$, as evidenced by the disparity in ϕ_{acc} between minority and majority demographic groups [251]. This effect varies with $\mathcal{M}^{dp}$'s architecture. Prior work has identified that training hyperparameters (e.g., gradient clipping and noise addition) disproportionately affect underrepresented classes [307, 306]. Furthermore, a long-tailed distribution results in poorer privacy protection for the data records belonging to the tail classes [250]. These data records tend to have higher gradient norms and lie closer to the decision boundary, which exacerbates the disparity [307]. Finally, lowering the privacy budget ϵ_{dp} (i.e., increasing privacy) worsens this disparity [251].

GpFair (Group Fairness)

● **Evsn (Evasion)** is easier against $\mathcal{M}^{gpfr}$. Training with fairness constraints reduces the distance of data records in $\mathcal{D}_{tr}$ to the decision boundary. This makes it easier to generate adversarial examples, increasing the susceptibility to Evsn [259].

● **Poisn (Poisoning)** is more effective against $\mathcal{M}^{gpfr}$ [309, 296, 310, 311]. These attacks increase disparity across demographic groups. There are various techniques to craft the poi-

soned data which includes leveraging $\mathcal{M}^{gpfr}$'s gradients [309], distorting the distribution of $\mathcal{D}_{tr}$ [296], skewing the decision boundary [311], and maximizing the covariance between sensitive attributes and classification labels [311].

● **`MemInf` (Membership inference)** is more effective against $\mathcal{M}^{gpfr}$ [258, 312]. Training with fairness constraints results in memorization of data records, especially from minority demographic groups. This reduces the distance of such data records to the decision boundary, thereby increasing their influence on model observables and making them more susceptible to `MemInf` [258]. Furthermore, `MemInf` increases with $\mathcal{M}^{gpfr}$'s capacity and the size of $\mathcal{D}_{tr}$, assuming that the fraction of minority groups remains constant [258]. These results are based on equalized odds as the fairness metric. It is not yet clear how this interaction changes under different fairness metrics (e.g., demographic parity), which remains an interesting future work (Chapter 9: Section 9.3.1).

● **`AttInf` (Attribute inference)** is less effective on $\mathcal{M}^{gpfr}$ due to the lower distinguishability in $\mathcal{M}^{gpfr}$'s predictions across demographic groups [256]. Prior work has shown that `GpFair`, with demographic parity as a constraint, can provably and empirically reduce `AttInf` [256]. On the contrary, Ferry et al. [313] show that sensitive attributes are easier to infer from $\mathcal{M}^{gpfr}$, suggesting increased susceptibility to `AttInf` (●). However, this result holds under a strong threat model in which $\mathcal{A}dv$ knows the group fairness algorithm and hyperparameters used to train $\mathcal{M}^{gpfr}$, allowing them to adapt their attack accordingly.

● **`Discr` (Discriminatory behaviour)** is lower for $\mathcal{M}^{gpfr}$ since it is designed to mitigate `Discr`. This can be achieved by *pre-processing* $\mathcal{D}_{tr}$ to remove bias [188]. Additionally, modifying the objective function to include a regularization term that penalizes fairness violations can reduce the distinguishability of $\mathcal{M}^{gpfr}$'s predictions across demographic groups, thereby reducing `Discr` [211, 196]. Finally, *post-training defenses* can help calibrate predictions to satisfy a given fairness metric [199].

`Expl` (Explanations)

● **`Evsn` (Evasion)** is easier since $\mathcal{A}dv$ can use `Expl` to generate adversarial examples that not only cause misclassification but also produce $\mathcal{A}dv$'s chosen explanation [317, 318]. Since `Expl` is computed using gradients that capture $\mathcal{M}$'s decision boundary, the information provided by `Expl` can facilitate the generation of adversarial examples.

● **`Poisn` (Poisoning)** is easier, as `Expl` provides a new attack surface where $\mathcal{M}$ can be manipulated to output $\mathcal{A}dv$-chosen explanations [319, 320, 321, 332]. Given the additional information provided by `Expl`, poisons can be crafted by shifting local optima and skewing $\mathcal{M}$'s parameters

and decision boundary [319, 321, 332, 322]. Explanations such as saliency maps and counterfactuals can similarly be manipulated [321, 322]. Prior work has shown that Poisn is more effective with larger model capacity and smoother curvature [320, 321, 322]. Finally, a smaller number of input attributes reduces the effectiveness of Poisn, as the information available to $\mathcal{A}dv$ through Expl is reduced.

● **ModExt (Unauthorized model ownership)** is easier, as Expl provides additional information about how different input attributes influence model predictions, capturing $\mathcal{M}$'s decision boundary [323, 324, 318]. This additional information can be used to train $\mathcal{M}^{sur}$. For instance, $\mathcal{A}dv$ can use counterfactuals obtained as output from $\mathcal{M}$ (i.e., minimally perturbed data records that change the prediction for a given input) to train $\mathcal{M}^{sur}$. The resulting $\mathcal{M}^{sur}$ has high ϕ_{acc} and fidelity, requiring fewer queries than when using only predictions from $\mathcal{M}$ [324]. The effectiveness of ModExt increases with $\mathcal{A}dv$'s dataset size and its distribution (e.g., imbalanced datasets) [324]. The number of queries to steal $\mathcal{M}$'s functionality can be reduced by using a "counterfactual of a counterfactual (as input to $\mathcal{M}$)" to train $\mathcal{M}^{sur}$ [323].

● **MemInf (Membership inference)** is easier as Expl leaks membership status of inputs: explanations reflect the distance of data records to the boundary which makes them distinguishable for records inside and outside $\mathcal{D}_{tr}$ [260, 325, 318, 326]. Further, the risk of MemInf decreases when there are fewer attributes as it information available to $\mathcal{A}dv$ from Expl is less. However, MemInf increases with $\mathcal{M}$'s capacity due to higher memorization [325].

● **DtRecon (Data reconstruction)** is easier on releasing Expl which capture $\mathcal{M}$'s sensitivity to different attribute values which allow for accurate reconstruction [327]. Hence, the risk of DtRecon increases with the quality of explanations. Moreover, some algorithms release influential $\mathcal{D}_{tr}$ data records for a given prediction on some input. This allows for perfect reconstruction of $\mathcal{D}_{tr}$ specially for data with a longer tail and with fewer attributes [260].

● **AttInf (Attribute inference)** is more effective, as Expl demonstrates distinguishability across different sensitive attributes, since it is specifically optimized to capture the influence of attributes on $\mathcal{M}$'s predictions [261]. This higher distinguishability across demographic groups is a potential contributor to the increased susceptibility to AttInf.

● **Discr (Discriminatory behaviour)** can be detected using Expl by identifying whether the relevant attributes are learned by $\mathcal{M}$ [208, 209]. This suggests that Expl reduces Discr (●). However, explanations are not perfect, i.e., they exhibit distinguishability across demographic groups, with disparities in the quality of Expl between minority and majority groups (●) [328, 329]. The quality of Expl varies with the number of attributes [328] and is more pronounced in models with larger capacity [329]. However, the underlying reasons for this behavior have not been systematically explored.

3.2.4 Guideline: Exploring Unintended Interactions

Based on the factors in the framework and their relation to defenses and risks, I will present a guideline to identify the nature of an unintended interaction and the underlying factors influencing it. The goal is to help practitioners make conjectures about the nature of the (unexplored) unintended interactions without requiring expensive empirical evaluation across multiple defenses and risks. Table 3.2 summarizes the statements about how a change in (i) the effectiveness of **Def** correlates with a change in <f>, and (ii) how a change in <f> correlates with the susceptibility to **Rk**. I use ↑ to identify positive correlation and ↓ for negative correlation. Hence, for a given **Def** and **Rk** with a common factor <f_c>, I infer how a change in effectiveness of **Def** correlates with **Rk** using the direction of the pair of arrows that describe how each of **Def** and **Rk** correspond to <f_c>.

Designing an exact algorithm for predicting unexplored interactions is difficult due to the intricate interplay of various factors. Therefore, I outline a guideline to make such conclusions with some expert knowledge about the nature of unintended interactions:

- For a given **Def** and **Rk**, identify <f_c>s from Table 3.2.
- For each <f_c>, if change in <f_c> on using **Def** aligns with change in **Rk**, i.e., (↑,↑) or (↓,↓), then effectiveness of **Def** corresponds to an increase in **Rk** (🔴); else **Def** negatively correlates with **Rk** (🟢) for (↑,↓) or (↓,↑).
- Consider the suggestions from all <f_c>s to infer the nature of the interaction. The conclusion is: (a) adopt the common suggestion when all <f_c>s agree; (b) if they conflict, follow the dominant <f_c>, identified using expert judgment.
- Empirically evaluate to see if other non-dominant <f>s influence the interaction. These additional factors can indicate conditions under which the nature of interaction will hold.

Choosing Dominant <f> with Expert Knowledge: Some factors, namely, O1, O2, O3, are directly exploited by different attacks. For instance, most (privacy) inference attacks rely on distinguishability in model observables (O2) while `Evsn` (evasion) and `Poisn` (poisoning) rely on curvature smoothness of the objective function (O1), and the distance of data records in $\mathcal{D}_{tr}$ to the decision boundary (O3). Using a defense directly correlated with a change in these factors has a significant effect on attack success. Hence, I identify them as dominant compared to other hyperparameter factors that implicitly influence the interactions, but are constant during the evaluation for a given experimental setup.

Evaluating the Proposed Guideline against Prior Work: To see how well the guideline covers different interactions, I compare conjectures from the guideline with corresponding observations in prior work as the ground truth (Section 3.2). I find that the guideline and the ground truth are consistent with four exceptions where the nature of the interaction changes depending on the threat model:

- EvsnRob→ ModExt,
- DiffPriv→ Evsn,
- DiffPriv→ DistInf, and
- GpFair→ AttInf.

The current guideline does not explain these exceptions, and extending the guideline to account for threat model is left as future work (discussed in Chapter 9: Section 9.3.1).

There are also cases where the guideline is not applicable as specific defenses and risks do not have common factors to identify the nature of some interactions. This includes interactions related to PoisnRob, Poisn, and ModExt where the correlation with various factors is not comprehensively explored. The specific interactions include

- EvsnRob→ Poisn, ModExt;
- PoisnRob→ Evsn, Poisn;
- DiffPriv→ Poisn;
- GpFair→ Poisn;
- Expl→ Evsn, Poisn, ModExt.

I also note that the guideline does not correctly capture the interactions of EvsnRob and DiffPriv with Discr. Here, the tail length of $\mathcal{D}_{tr}$'s distribution (D2) is a common factor. However, for EvsnRob and DiffPriv, D2 refers to the distribution over different classes rather than demographic groups (e.g., [237]). This difference in the semantics leads to an incorrect conjecture. One possible approach is to split the factors into more precise categories. For example, separate factors could capture the tail length of the $\mathcal{D}_{tr}$ distribution across classes and demographic groups. This would remove D2 as the common factor, and will remove the incorrect conjecture.

3.3 Unexplored Interactions and Conjectures

The natural question is whether one can use the guideline to conjecture about unexplored interactions. I enumerate different unexplored interactions in Table 3.5, which are marked as ● in Tables 3.3 and 3.4. To demonstrate the applicability of the guideline, I select three examples for empirical evaluation among the unexplored interactions in Table 3.4 based on ease of implementation[2]: (i) group fairness (GpFair) with data reconstruction (DtRecon) in Section 3.3.1; (ii) group fairness (GpFair) with distribution inference (DistInf) in Section 3.3.2; and (iii) explanations (Expl) with distribution inference (DistInf) in Section 3.3.3. I apply

[2]Code: https://github.com/ssg-research/sok-unintended-interactions.

the guideline to conjecture about the nature of interactions for these three unexplored cases and validate them empirically. I mark the factors evaluated for these interactions with $\ast$ in Table 3.4. These examples are only intended for illustrating the applicability of the guideline. I leave a systematic empirical exploration of remaining unexplored interactions as future work. In Section 3.3.4, I speculate the remaining unexplored interactions.

Table 3.5: **Summary of unexplored interactions:** These interactions are marked as ● in Table 3.3 and 3.4, and the interaction between a defense to a risk with "→" between them.

Defense → Risk
`EvsnRob` (Adversarial Training) → `AttInf` (Attribute Inference)
`PoisnRob` (Outlier Removal) → `ModExt` (Unauthorized Model Ownership) `PoisnRob` (Outlier Removal) → `DtRecon` (Data Reconstruction) `PoisnRob` (Outlier Removal) → `DistInf` (Distribution Inference)
`MdlWM` (Watermarking) → `Evsn` (Evasion)
`Fngrprnt` (Fingerprinting) → `Poisn` (Poisoning) `Fngrprnt` (Fingerprinting) → `DtRecon` (Data Reconstruction) `Fngrprnt` (Fingerprinting) → `AttInf` (Attribute Inference) `Fngrprnt` (Fingerprinting) → `DistInf` (Distribution Inference) `Fngrprnt` (Fingerprinting) → `Discr` (Discriminatory Behaviour)
`DiffPriv` (Differential Privacy) → `ModExt` (Unauthorized Model Ownership)
`GpFair` (Group Fairness) → `ModExt` (Unauthorized Model Ownership) `GpFair` (Group Fairness) → `DtRecon` (Data Reconstruction) `GpFair` (Group Fairness) → `DistInf` (Distribution Inference)
`Expl` (Explanations) → `DistInf` (Distribution Inference)

3.3.1 `GpFair` (Group Fairness) → `DtRecon` (Data Reconstruction)

Conjecture: From Table 3.2, I identify distinguishability in model observables across demographic groups (O2.2) as a common factor (<f_c>). As the arrows are in opposite directions, I conjecture that `GpFair` reduces `DtRecon` →●. Now, I check for other factors: the number of input attributes (D3) suggests that `DtRecon` is less effective when there are more input attributes. Thus, I expect that the decrease due to `GpFair` is only evident when the number of attributes is small where `DtRecon` is still effective. Consequently, the difference in attack success between $\mathcal{M}^{gpfr}$ and $\mathcal{M}$ becomes negligible for more attributes. I describe the experimental setup below.

Threat Model: I assume x is sensitive and confidential. $\mathcal{A}dv$ only observes $\mathcal{M}^{gpfr}(x)$ for some

unknown confidential x that is to be reconstructed. $\mathcal{A}dv$ also has access to an auxiliary dataset $\mathcal{D}^{aux}$ that is split into $\mathcal{D}^{aux}_{tr}$ to train an attack decoder model (f_{att}) and $\mathcal{D}^{aux}_{te}$ for evaluation.

Attack Methodology: $\mathcal{A}dv$ uses $\mathcal{D}^{aux}_{tr}$ to train f_{att} to map $\mathcal{M}^{gpfr}(x)$ to x, minimizing the error between $f_{att}(\bar{\mathcal{M}}^{fair}(x))$ and the ground truth x. Here, $\bar{\mathcal{M}}^{fair}$ is a "shadow model" which is trained on $\mathcal{A}dv$'s auxiliary dataset to have similar functionality as $\mathcal{M}^{gpfr}$. I give the advantage to $\mathcal{A}dv$ and assume $\mathcal{M}^{gpfr} == \bar{\mathcal{M}}^{fair}$. The attack is evaluated on $\mathcal{D}^{aux}_{te}$.

Experimental Setup: I use the CENSUS dataset which consists of attributes like age, race, education level, to predict whether an individual's annual income exceeds 50K. I select the top 10 attributes and consider both race and sex attributes. I use $|\mathcal{D}_{tr}|$=15470, $|\mathcal{D}^{aux}_{tr}|$=7735, and $|\mathcal{D}^{aux}_{te}|$=7735. I train $\mathcal{M}^{gpfr}$ with adversarial debiasing which uses an adversarial network f_{inf} to infer sensitive attribute values from $\mathcal{M}^{gpfr}(x)$ [211]. The loss from f_{inf} is used to train $\mathcal{M}^{gpfr}$ such that $\mathcal{M}^{gpfr}(x)$ is indistinguishable. I measure fairness by (a) checking if p%-rule is $> 80\%$ as required by the regulations [333], and (b) AUC score of f_{inf} to correctly infer the sensitive attribute values (fair model should have a score of 0.50). To measure attack success, I use the reconstruction loss (l_2-norm) between x and $f_{att}(\bar{\mathcal{M}}^{fair}(x))$ (the higher the loss, the less effective the attack). I use a multi-layered perceptron (MLP) for $\mathcal{M}^{gpfr}$ with hidden layer dimensions [32, 32, 32, 1], f_{inf} with dimensions [32, 32, 32, 2], and f_{att} with dimensions [18, 32, 64, 128, 10]. I compare the attack success with the baseline of data reconstruction against $\mathcal{M}$ with the same architecture as $\mathcal{M}^{gpfr}$ but without group fairness.

Empirical Validation: I validate the conjecture by showing the effect on distinguishability across demographic groups (O2.2), and evaluating the influence of the number of input attributes (D3).

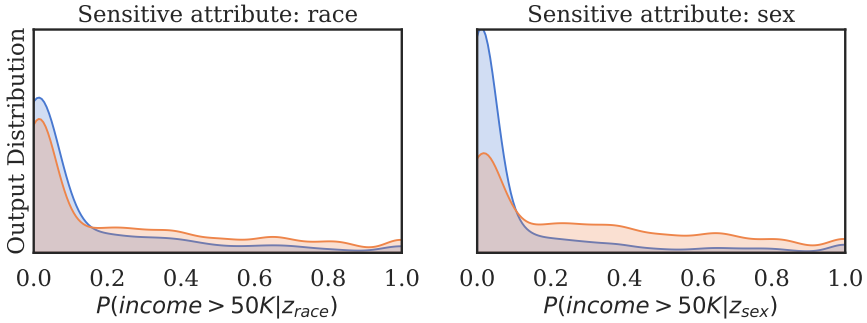
Nature of Interaction: I compare the reconstruction loss between $\mathcal{M}^{gpfr}$ and $\mathcal{M}$ in Table 3.6. I confirm that training with adversarial debiasing is indeed fair (p%-rule $> 80\%$ and AUC score to predict attributes ~ 0.50). I observe that $\mathcal{M}^{gpfr}$ has a higher reconstruction loss. Hence, this validates the conjecture about the nature of the interaction.

Distinguishability in model observables across demographic groups (O2.2): Figure 3.1 compared $\mathcal{M}^{gpfr}$'s prediction distribution with $\mathcal{M}$ for the two attributes: race (whites as red vs non-whites as blue) and sex (males as red vs females as blue). We can see that there is no income disparity between whites-non-whites and males-females for $\mathcal{M}^{gpfr}$. Further, $\mathcal{M}^{gpfr}$'s predictions are normalized which reduces the information available to $\mathcal{A}dv$ for successful reconstruction.

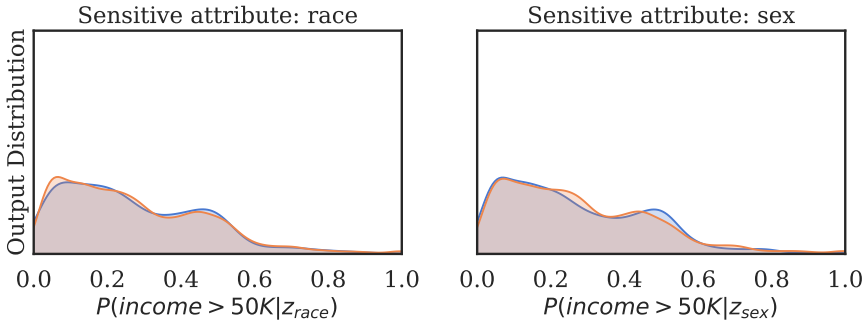
Number of Input Attributes (D3): I present the impact of the number of input attributes in Table 3.7. Firstly, we can note that ϕ_{acc} is similar across different number of input attributes for $\mathcal{M}^{gpfr}$ regardless of the difference in reconstruction loss. Hence, ϕ_{acc} does not impact the

Table 3.6: **Nature of interaction:** Comparison of ϕ_{acc} on $\mathcal{D}_{te}$, fairness (AUC scores, p%-rule values) and reconstruction loss for $\mathcal{M}$ and $\mathcal{M}^{gpfr}$ across ten runs. $\Uparrow$ indicates higher value is desirable and $\Downarrow$ indicates lower value is desirable.

Metric	$\mathcal{M}$	$\mathcal{M}^{gpfr}$
Acc $\Uparrow$	84.40 $\pm$ 0.09	77.96 $\pm$ 0.58
AUC $\Downarrow$	00.66 $\pm$ 0.00	00.52 $\pm$ 0.01
p%-rule **Race** $\Uparrow$	46.51 $\pm$ 0.59	93.69 $\pm$ 5.60
p%-rule **Sex** $\Uparrow$	29.70 $\pm$ 1.15	90.50 $\pm$ 6.87
ReconLoss $\Uparrow$	00.85 $\pm$ 0.01	00.95 $\pm$ 0.02



(a) $\mathcal{M}$'s Prediction Distribution



(b) $\mathcal{M}^{gpfr}$'s Prediction Distribution

Figure 3.1: **Distinguishability of model observables across demographic groups** (02.2) decreases when fairness constraints are applied $\mathcal{M}^{gpfr}$.

risk, and only the number of input attributes is the variable. I present the reconstruction loss for a different number of input attributes $\{10, 20, 30\}$. As expected, for 10 attributes, group fairness reduces attack success significantly (as also seen in Table 3.6). However, increasing the

Table 3.7: **Number of input attributes (D3):** Reconstruction loss and ϕ_{acc} on $\mathcal{D}_{te}$ for different number of input attributes for $\mathcal{M}$ and $\mathcal{M}^{gpfr}$ (averaged across ten runs).

# Input Attributes	$\mathcal{M}$		$\mathcal{M}^{gpfr}$	
	ReconLoss	Acc	ReconLoss	Acc
10	0.85 ± 0.00	84.40 ± 0.09	0.95 ± 0.02	78.96 ± 0.58
20	0.93 ± 0.03	84.72 ± 0.22	0.93 ± 0.00	80.32 ± 1.12
30	0.95 ± 0.02	84.41 ± 0.39	0.94 ± 0.00	79.50 ± 0.91

number of input attributes has no significant difference in reconstruction loss between $\mathcal{M}$ and $\mathcal{M}^{gpfr}$, thus, validating the conjecture.

Takeaway: The conjectures from the guideline about the interaction between `GpFair` and `DtRecon`, including the conditions under which it occurs, are successfully validated.

3.3.2 `GpFair` (Group fairness) → `DistInf` (Distribution Inference)

Conjecture: I identify the distinguishability of model observables across demographic groups (`O2.2`) as <f$_c$>. Since the arrows point in opposing directions, this suggests that `GpFair` reduces susceptibility to `DistInf` by decreasing the distinguishability of model observables across demographic groups (→●).

Threat Model: I assume that $\mathcal{A}dv$ has whitebox access to $\mathcal{M}$ and the corresponding $\mathcal{M}^{gpfr}$ trained with fairness constraints. This allows $\mathcal{A}dv$ to mount a whitebox `DistInf` [12].

Attack Methodology: $\mathcal{A}dv$ trains a binary *attack classifier* (f_{att}) on the first-layer parameters of models to distinguish those trained on datasets with ratios α_1 and α_2 (e.g., 50% vs. 10% women in $\mathcal{D}_{tr}$), using the model predictions. $\mathcal{A}dv$ then compares the attack accuracy of f_{att} between $\mathcal{M}$ and $\mathcal{M}^{gpfr}$ to assess how training with fairness constraints impacts `DistInf`.

Experimental Setup: I use CENSUS and UTKFACE datasets. CENSUS is a tabular dataset with 15 attributes used to predict whether income exceeds $50K. I use 2,000 training samples and 500 test samples for $\mathcal{D}_{tr}$, and 1,100 training samples and 500 test samples for $\mathcal{D}^{aux}$. I use a fully connected neural network with hidden layers [1024, 512, 256, 2] and $\tanh$ activations. UTKFACE consists of 48×48 RGB facial images for binary classification of individuals as young (under 30). I use 2,000 training samples and 500 test samples for $\mathcal{D}_{tr}$, and 1,100 training samples and 500 test samples for $\mathcal{D}^{aux}$, using a VGG16 model. I use race and sex as sensitive attributes for CENSUS, and only sex for UTKFACE.

Following prior work [12], I fix $\alpha_1 = 0.5$ and vary α_2={0.1-0.4,0.6-0.9}. I measure the attack effectiveness using f_{att}'s accuracy on 100 unseen (test) models. I use equalized odds to quantify bias: values near zero indicate low bias (more fair), while higher values indicate significant bias (poor fairness) [200]. To obtain $\mathcal{M}^{gpfr}$, I train using a fairness constraint as a regularizer to minimize equalized odds, while $\mathcal{M}$ is trained without this fairness regularization.

Table 3.8: **Nature of interaction (`Discr` $\rightarrow$ `DistInf`)**: I report the attack accuracy where $\alpha_1 = 0.5$ and vary α_2. I compare the attack accuracy on $\mathcal{M}$ (as baseline) to $\mathcal{M}^{gpfr}$, where a value close to 50% indicates random guess accuracy.

α_2	CENSUS (race)		CENSUS (sex)		UTKFACE (sex)	
	$\mathcal{M}$	$\mathcal{M}^{gpfr}$	$\mathcal{M}$	$\mathcal{M}^{gpfr}$	$\mathcal{M}$	$\mathcal{M}^{gpfr}$
0.1	$100.00_{\pm 0.00}$	$50.00_{\pm 0.00}$	$59.80_{\pm 7.02}$	$50.20_{\pm 0.45}$	$66.30_{\pm 1.44}$	$50.10_{\pm 0.22}$
0.2	$98.20_{\pm 0.27}$	$50.00_{\pm 0.00}$	$52.20_{\pm 2.14}$	$50.00_{\pm 0.00}$	$70.00_{\pm 0.50}$	$50.30_{\pm 0.27}$
0.3	$76.90_{\pm 3.52}$	$50.00_{\pm 0.00}$	$98.30_{\pm 1.25}$	$50.00_{\pm 0.00}$	$68.10_{\pm 2.04}$	$53.40_{\pm 0.96}$
0.4	$49.80_{\pm 0.45}$	$50.00_{\pm 0.00}$	$69.90_{\pm 5.16}$	$50.00_{\pm 0.00}$	$57.10_{\pm 0.22}$	$50.00_{\pm 0.00}$
0.6	$49.40_{\pm 0.42}$	$51.30_{\pm 2.64}$	$86.10_{\pm 1.64}$	$50.00_{\pm 0.00}$	$73.00_{\pm 0.61}$	$50.00_{\pm 0.00}$
0.7	$51.10_{\pm 2.38}$	$50.00_{\pm 0.00}$	$73.40_{\pm 2.58}$	$50.00_{\pm 0.00}$	$88.40_{\pm 0.22}$	$50.00_{\pm 0.00}$
0.8	$87.80_{\pm 3.17}$	$50.00_{\pm 0.00}$	$52.90_{\pm 2.86}$	$50.00_{\pm 0.00}$	$88.10_{\pm 0.65}$	$50.00_{\pm 0.00}$
0.9	$93.90_{\pm 1.67}$	$50.00_{\pm 0.00}$	$51.10_{\pm 2.38}$	$50.00_{\pm 0.00}$	$86.70_{\pm 0.27}$	$50.00_{\pm 0.00}$

Results: Table 3.8 shows `DistInf` attack accuracy ("Acc") across different ratios and datasets. I omit equalized odds values, as $\mathcal{M}$ consistently exhibits higher values than $\mathcal{M}^{gpfr}$ across all ratios, as expected. I use the attack accuracy on $\mathcal{M}$ as the baseline and compare it with the accuracy on $\mathcal{M}^{gpfr}$ to assess whether the attack degrades. In other words, lower attack accuracy for $\mathcal{M}^{gpfr}$ indicates that `GpFair` reduces susceptibility to `DistInf`.

Across all ratios and datasets, the attack accuracy for $\mathcal{M}^{gpfr}$ is significantly lower (close to random-guess accuracy of $\sim 50\%$), compared to $\mathcal{M}$ ($> 70\%$), where the attack is effective. For four ratios in Census (race and sex), the accuracies are statistically similar; however, they remain close to the random-guess baseline of 50%. Overall, these results indicate that `GpFair` reduces susceptibility to `DistInf` as conjectured.

Takeaway: The conjecture from the guideline about the nature of the interaction between `GpFair` and `DistInf`, is successfully validated.

3.3.3 `Expl` (Explanations) → `DistInf` (Distribution Inference)

Conjecture: I identify the distinguishability of model observables across datasets (O2.1) as <f_c>. As the arrows are the same direction, I conjecture that releasing `Expl` can result in `DistInf` (●). Specifically, `Expl` demonstrates distinguishability across different datasets, which extends to datasets with differing distributional properties. This distinguishability enables successful `DistInf`. I also check for other factors: the number of input attributes (D3) and model capacity (M1). For D3, increasing the number of input attributes should reduce `DistInf` by making it harder to capture distributional properties. For M1, increasing $\mathcal{M}$'s capacity should increase memorization of distributional properties, thereby increasing `DistInf`. I describe the experimental setup below to validate these conjectures.

Threat Model: I assume that $\mathcal{M}$ outputs both explanations $\gamma(x)$ and $\mathcal{M}(x)$ corresponding to an input x. $\mathcal{A}dv$ differentiates between $\mathcal{M}$ trained on $\mathcal{D}_{tr}$ with ratio α_1 vs. α_2 (e.g., 50% vs. 10% women in $\mathcal{D}_{tr}$) by distinguishing between their respective explanations $\gamma(x)^{\alpha_1}$, and $\gamma(x)^{\alpha_2}$ by training an ML attack model f_{att}. $\mathcal{A}dv$ has an auxiliary dataset which is split into $\mathcal{D}^{aux}_{tr}$ and $\mathcal{D}^{aux}_{te}$ to train and evaluate f_{att}.

Attack Methodology: Given two ratios, α_1 or α_2, $\mathcal{A}dv$ samples $\mathcal{D}^{aux}_{tr}$ to get ${}^{\alpha_1}\mathcal{D}^{aux}_{tr}$ and ${}^{\alpha_2}\mathcal{D}^{aux}_{tr}$ satisfying α_1 or α_2. $\mathcal{A}dv$ then trains multiple *shadow models* on these datasets which try to mimic $\mathcal{M}$; $\mathcal{A}dv$ uses $\gamma'(x)^{\alpha_1}$ and $\gamma'(x)^{\alpha_2}$ from the shadow models to train f_{att}. Given $\mathcal{M}$ which is trained on $\mathcal{D}_{tr}$ satisfying some unknown ratio, $\mathcal{A}dv$ uses $\gamma(x)$ where $x \in \mathcal{D}^{aux}_{te}$ from $\mathcal{M}$ as input to f_{att} to infer if $\mathcal{D}_{tr}$'s property was α_1 or α_2.

Setup: I use the CENSUS dataset from Section 3.3.1 with 42 input attributes. I consider different ratios of women in $\mathcal{D}_{tr}$ as the property of interest in the range $0.1 - 0.9$ incremented by 0.1. Following prior work [12, 121], I set $\alpha_1 = 0.5$ while varying α_2. For each value of α_2, $\mathcal{A}dv$ trains f_{att} to differentiate between $\alpha_1 = 0.5$ and the specific value of α_2. I measure the attack success using (distribution) inference accuracy, where 50% is random guess. I use an MLP with the following hidden layer dimensions: [1024, 512, 256, 128]. I use three explanation algorithms: IntegratedGradients [205], DeepLift [334, 335], and SmoothGrad [217]. For explanations, I want to evaluate if releasing explanations increases `DistInf`. Following the approach by Pawelczyk et al. [325], assuming random guess (50%) as the baseline, I want to show that an attack that *only* makes use of explanations can do better than the baseline. The aim is to determine if explanations leak distribution information, and not whether explanations can improve the state-of-the-art distribution inference attacks.

Empirical Validation: I validate the conjecture and evaluate how it is influenced by the number of input attributes (D3), and model capacity (M1).

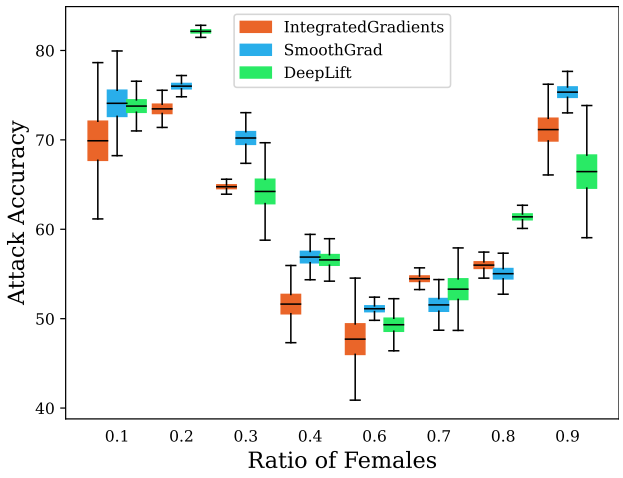


Figure 3.2: **Nature of interaction (`Expl` $\rightarrow$ `DistInf`):** Attack accuracy of `DistInf` to differentiate explanations from model trained on $\mathcal{D}_{tr}$ with α_1=0.5 and $\alpha_2 \in \{0.1 - 0.9\}$.

Nature of Interaction: I present the attack accuracy across different ratios and explanation algorithms in Figure 3.2. I use all 42 attributes for evaluation. For the three explanation algorithms, the attack accuracy is higher than a random guess of 50% for most of the ratios, validating the conjecture that explanations indeed leak distribution properties of $\mathcal{D}_{tr}$, and increase the susceptibility to `DistInf`.

Number of Input Attributes (D3): I use an attribute selection algorithm to select the top 35, 25 and 15 attributes from the original 42 attributes. I evaluate the impact of varying the top-k relevant attributes on the interaction. I fix α_1=0.5 and α_2=0.1 as it shows the highest attack accuracy (Figure 3.2). I present the results in Table 3.9.

Table 3.9: **Number of input attributes (D3):** Attack accuracy for $\alpha_1 = 0.5$ and $\alpha_2 = 0.1$ while varying the number of input attributes (results averaged across ten runs).

# Attributes	IntegratedGradients	DeepLift	SmoothGrad
15	81.07 $\pm$ 2.13	78.74 $\pm$ 1.66	65.40 $\pm$ 1.39
25	66.09 $\pm$ 0.95	73.64 $\pm$ 1.38	59.42 $\pm$ 1.09
35	50.43 $\pm$ 0.59	59.93 $\pm$ 2.81	56.78 $\pm$ 1.93

I confirm that increasing the number of most relevant input attributes decreases the attack success: a greater number of attributes decreases the memorization of each attribute thereby reducing the quality of explanations.

Model Capacity (M1): I consider four MLPs with different model capacities: "Model 1" with dimensions [128], "Model 2" with [256, 128], "Model 3" with [256,512,256], and "Model 4" with dimensions [1024, 512, 256, 128]. I present the attack accuracy in Table 3.10.

Table 3.10: **Model capacity (M1):** Attack accuracy on explanations corresponding to α_1=0.5 and α_2=0.1 for different models (averaged across ten runs).

# Parameters	IntegratedGradients	DeepLift	SmoothGrad
Model 1 (5762)	47.57 ± 4.25	49.19 ± 2.75	53.26 ± 0.10
Model 2 (44162)	53.29 ± 3.65	50.86 ± 3.24	62.40 ± 0.95
Model 3 (274434)	62.60 ± 2.74	67.73 ± 1.69	70.21 ± 0.73
Model 4 (733314)	69.90 ± 3.24	73.78 ± 1.03	74.09 ± 2.17

All models have an accuracy of $82.10 \pm 1.17\%$ and close to perfect $\mathcal{D}_{tr}$ accuracy. Hence, neither accuracy nor $\mathcal{G}_{err}$ impact `DistInf`. I observe that `DistInf` increases with capacity: a higher capacity correlates with greater memorization of $\mathcal{D}_{tr}$'s distributional properties.

Takeaway: The conjectures from the guideline about the interaction between `Expl` and `DistInf`, including the conditions under which it occurs, are successfully validated.

3.3.4 Other Unintended Interactions

I reflect on other unintended interactions.

`EvsnRob` (Adversarial Training) and `AttInf` (Attribute Inference)

The tail length of $\mathcal{D}_{tr}$'s distribution (D2) and the priority of learning stable attributes (D4) are the common factors. As the arrows are in the opposite directions, it suggests a decrease in `AttInf` (●). However, distinguishability of model observables across demographic groups (O2.2) is likely to be the dominant factor. Hence, the interaction will be determined by whether the distinguishability in model observables across demographic groups increases or decreases with `EvsnRob`.

`PoisnRob` (Poison Robustness) and `DtRecon` (Data Reconstruction)

The size of $\mathcal{D}_{tr}$ (D1) is the common factor and the arrows are in the same direction, suggesting an increase in `DtRecon` (●). Similar to `MemInf` and `AttInf` [240, 76, 115], I expect retraining

without outliers will make $\mathcal{M}^{psnrob}$ memorize a new set of data records, thus making them more susceptible to DtRecon.

PoisnRob (Poison Robustness) and DistInf (Distribution Inference)

The tail length of $\mathcal{D}_{tr}$'s distribution (D2) is the common factor which has arrows in the same direction which suggests an increase in DistInf (●). Prior work has shown that adding more outliers is likely to increase DistInf [274, 275]. PoisnRob removes outliers from tail classes. Assuming that these outliers belong to the minority group, and PoisnRob could increase the bias thereby increasing the distinguishability in model observables for datasets with different properties. Hence, DistInf is likely to increase.

3.4 Discussions

Completeness of Framework: I identified overfitting and memorization (and their underlying factors) as the underlying causes of unintended interactions based on prior work. These are sufficient to explain all the unintended interactions explored in the literature as seen in the survey (Sections 3.1 and 3.2). I do not claim them to be complete considering the complexity of ML models. Whether there are indeed other possible underlying causes, remains an open question. Regardless, the framework is flexible enough to accommodate emerging causes and factors. To incorporate a new underlying cause, one can use the same methodology as in Section 3.1 and 3.2: identify factors that influence the cause, describe how they relate to defenses and risks, and use them to conjecture about unintended interactions. Further, the framework can be extended to cover variants of current risks and defenses as well as new ones by expanding Tables 3.2, 3.3, and 3.4. Finally, the framework can be extended to account for any exceptions in some interactions stemming from a difference in threat model. I describe a possible approach later, in Chapter 9: Section 9.3.

Minimizing Unintended Interactions: Given the factors influencing unintended interactions, practitioners can mitigate them by exploring modifying individual factors: (i) data augmentation (D1), (ii) balanced classes with undersampling and oversampling (D2), (iii) adding proxy attributes or removing redundant attributes (D3), (iv) domain generalization algorithms (e.g., invariant risk minimization) [91, 229] (D4), (v) adjusting training hyperparameters (O1, O2, and O3), and (vi) experimenting with more layers, lower precision, or pruning (M1). However, modifying these factors will reduce specific unintended interactions, and is not a general solution. An important open problem is how to combine multiple defenses effectively to protect against multiple risks. I discuss this in Chapter 4.

Applicability beyond classifiers: The framework can be extended to other models by: (i) identifying unintended interactions, (ii) evaluating the relation between defenses, risks, and underlying factors, and (iii) including them in the framework for insights using the guideline. Recent work by Meng et al. [336] builds on this work to evaluate unintended interactions among defenses and unrelated risks in language models. However, they empirically evaluated the unintended interactions without a discussion on overfitting and memorization. To apply the framework to generative models, some factors influencing overfitting and memorization, such as size of the training data (D1) and capacity of $\mathcal{M}$ (M1), are still relevant [72, 86, 87]. However, there will be additional factors specific to generative models that are not applicable to classifiers (e.g., probability distribution to sample outputs). Furthermore, there are factors specific to classifiers that are not applicable for generative models (e.g., distance to decision boundary). Hence, while several factors in the framework as well as the guideline, are applicable to generative models, a systematic study to extend the proposed framework to generative models is an interesting direction for future work (Chapter 9: Section 9.3).

Library for Evaluating Unintended Interactions: I have also contributed to the design of *Amulet*, the first Python library for evaluating both intended and unintended interactions among ML defenses and risks [63]. Amulet is *comprehensive* by including the representative attacks, defenses, and metrics, *extensible* to new modules due to its modular design, *consistent* with a user-friendly API template for risks, defenses, and metrics, and *applicable* for evaluating novel unintended interactions. Amulet offers a unified framework to evaluate unintended interactions among defenses and unrelated risks.

Contribution in Support of the Thesis Statement

Existing work lacks a comprehensive understanding of unintended interactions between defenses and unrelated risks, which is crucial for real-world deployment of ML models. Based on the conjecture that overfitting and memorization are the underlying causes, I identify various factors influencing overfitting and memorization covering properties of the training data (e.g., size, distribution), objective function (e.g., distance to boundary, loss curvature), and the model (e.g., capacity). These factors enable a systematic analysis and prediction of unintended interactions, answering **RQ1.1** and supporting the thesis statement.

Chapter 4

Combining ML Defenses without Conflicts

Overview

Real-world models need simultaneous protection against multiple risks. This necessitates combining multiple defenses *effectively*, without incurring significant drop in the effectiveness of the constituent defenses. I systematize existing work based on *how the defenses are combined* and *how they interact when they are combined*. I use this to identify unexplored combinations and the limitations of existing combination techniques. Using these insights, I present `Def\Con`, a combination technique which is (a) *accurate* (correctly identifies whether a combination is effective or not), (b) *scalable* (allows combining multiple defenses), (c) *non-invasive* (allows combining existing defenses without modification), and (d) *general* (is applicable to different types of defenses). I show that `Def\Con` achieves 90% balanced accuracy on eight combinations from prior work, and 86% on 30 unexplored combinations that I evaluated empirically. Overall, by using the factors[a] underlying conflicting interactions among defenses, I show that it is possible to effectively combine defenses, thus, answering **RQ1.2** to support the thesis statement.

[a] **Terminology:** Unlike in Chapter 3, where the factors relate to overfitting and memorization, the factors here concern interference among defenses.

Real-world deployment of machine learning (ML) models requires practitioners to account and protect against multiple risks simultaneously. But defenses designed to protect against one risk, may impact susceptibility to other unrelated risks (Chapter 3). This raises the question of how to combine defenses *effectively*, without incurring a significant drop in the protection

provided by each constituent defense. Practitioners need effective *combination techniques*, either by *modifying existing defenses* (invasive), or by identifying whether *existing defenses can be combined without modification* (non-invasive). Prior work either explores conflicting interactions among defenses but is limited to specific defenses (e.g., [236, 221, 231, 50]), or studies interactions among defenses with risks (e.g., Chapter 3 and [219, 336]). None of the prior work provides a principled way to quickly determine if *defenses can be combined effectively*.

I aim to answer the following research question: (**RQ1.2**) *how can we effectively combine ML defenses while avoiding conflicts?* To this end, I first systematically survey existing work on combining defenses based on: (a) *how the defenses are combined* (i.e., what combination technique was used), and (b) *how the defenses interact* (i.e., whether they conflict or align). I identify previously unexplored combinations, and evaluate prior combination techniques to identify their limitations and inform the design of a new combination technique. Among prior combination techniques, non-invasive techniques are easier to deploy as they do not require expert knowledge from practitioners. Therefore, I identify an existing non-invasive technique ("*mutually exclusive placement*" [337]) as promising. It presumes that two defenses can be effectively combined iff they *operate on different stages in the ML pipeline*: pre-, in-, and post-training. However, it can result in ineffective combinations: (a) later-stage defenses can conflict with earlier ones (e.g., model or dataset watermarking with adversarial training or differential privacy [236]), and (b) same-stage defenses may still be compatible (see Section 4.5).

Based on these insights, I present `Def\Con`, a combination technique which is (i) *accurate* (correctly identifies whether a combination is effective or not), (ii) *scalable* (allows two or more defenses to be combined), (iii) *non-invasive* (does not require changes to the defenses being combined), and (iv) *general* (applicable to different types of defenses). `Def\Con` is inspired by mutually exclusive placement (aka naïve technique), but overcomes its limitations by explicitly accounting for the factors that underlie conflicts among defenses: a later-stage defense either (a) mitigates a risk re-purposed as a defense by an early-stage defense, or (b) overrides changes made by an early-stage defense [236]. I claim the following contributions:

1. a *systematization* of prior work on combination techniques based on *how the defenses are combined* and *how the defenses interact* (i.e., objectives conflict or align); (Section 4.1)
2. *identify* unexplored combinations and the limitations of prior techniques to design `Def\Con`[1], a scalable, non-invasive, and general combination technique; (Section 4.2 and 4.3)
3. *comprehensive evaluation* to show that `Def\Con` is more accurate than the naïve technique (Section 4.4 and 4.5), with a balanced accuracy of
   - 90% (`Def\Con`) vs. 40% (naïve) using eight combinations from prior work as ground truth,
   - 86% (`Def\Con`) vs. 36% (naïve) via empirical evaluation of 30 unexplored combinations.

[1]Link to Code: https://github.com/ssg-research/combining-defenses.

`Def\Con` is an inexpensive and fast technique for practitioners to determine if a combination of existing defenses can be effective, without expensive empirical evaluation.

4.1 Systematization of Related Work

I present the methodology for systematization followed by survey of prior work based on *how the defenses are combined*, and *how they interact when they are combined*. I select the defenses from Chapter 2: Section 2.4 and explore their combinations.

Methodology for Systematization

I enumerate all defense combinations from Chapter 2: Section 2.4 in Table 4.1. Each combination is represented as a cell in Table 4.1, for which I indicate related work, combination technique used, and the nature of resulting interaction. For each prior work, I identify the following: (i) **combination technique** where I indicate the technique used to combine defenses as *optimization* (**T1**) or *mutually exclusive placement* (**T2**); and (ii) **nature of interaction** where I indicate the interaction among defenses as a *conflict* (●), *alignment* (●), or *unexplored* (●). I describe the details below.

Combination Techniques

I present two combination techniques based on prior work which either *modify existing defenses*, or *identify whether existing defenses can be combined without modification*. I mark them as **T1** and **T2** respectively, and describe them below:

T1 Optimization includes game-theoretic formalization, regularization, or constrained equation solving. **T1** incorporates defenses into the objective function (e.g., regularization terms) so that the corresponding defense constraints can be satisfied during the training for an effective combination [338, 339, 340, 341, 342, 343, 259, 344, 61, 345, 346, 347, 348, 349, 350, 351, 352, 353, 295, 354, 355, 356, 357, 358, 307, 359, 360, 361, 362, 306, 363, 364, 365, 366, 367]. This also includes using variants of the standard model architectures and algorithms, specifically catered for a particular combination to give better trade-offs among the defenses [360, 361, 368, 369, 370].
T2 Mutually Exclusive Placement consists of applying defenses at *different* stages of the ML pipeline— (i) pre-training (modifies $\mathcal{D}_{tr}$), (ii) in-training (modifies training configuration such as the objective function), (iii) post-training (modifies the inputs to or outputs of trained $\mathcal{M}$

during inference)—to avoid conflicts [337, 371]. I later refer to **T2** as the *naïve technique* since I use it as a baseline to compare with the proposed technique, `Def\Con` (Section 4.3 and 4.5).

Nature of Interaction

Consider two defenses $\texttt{Def}_1$ and $\texttt{Def}_2$ which protect against the risks $\texttt{Rk}_1$ and $\texttt{Rk}_2$ respectively, where $\texttt{Def}_2$ is applied after $\texttt{Def}_1$. The defenses can interact in one of two ways:

- **Alignment:** $\texttt{Def}_1$ and $\texttt{Def}_2$ are *aligned* if any of these hold: (i) $\texttt{Def}_1$ and $\texttt{Def}_2$ do not impact $\texttt{Rk}_2$ and $\texttt{Rk}_1$, respectively; (ii) $\texttt{Def}_1$ reduces $\texttt{Rk}_2$, increasing $\texttt{Def}_2$'s effectiveness; and (iii) $\texttt{Def}_1$ generalizes $\texttt{Def}_2$, so its effectiveness implies that of $\texttt{Def}_2$. Alignment leads to an effective defense combination. When one defense implies the other (as in case (iii)), applying the first defense may be sufficient since the second is free (e.g., `GpFair` and resistance to `AttInf` [256]).
- **Conflict:** $\texttt{Def}_1$ and $\texttt{Def}_2$ *conflict* if any of the following hold: (i) $\texttt{Def}_1$ uses risk $\texttt{Rk}$ (protected by $\texttt{Def}_2$), making $\texttt{Def}_1$ ineffective; and (ii) $\texttt{Def}_2$ overrides $\texttt{Def}_1$'s changes, making $\texttt{Def}_1$ ineffective. Conflict leads to an ineffective combination of the defenses. To avoid conflicts, accurate combination techniques are needed. I refer to these as the factors related to the interference among defenses.

Justification for Systematization: Prior surveys on conflicts among defenses are limited to specific defenses (e.g., [221, 231, 50]), or lack sufficient details about the combination techniques used (e.g., [219, 372]). This makes it challenging to design better combination techniques, and systematically compare with prior work. The presented systematization addresses these limitations by (a) covering multiple defenses and their combinations, and (b) explicitly mapping the combinations to the corresponding techniques and the nature of interactions among defenses. As shown later in Section 4.2, the systematization helps to identify gaps in existing literature (e.g., unexplored combinations, and limitations of prior techniques). These insights will inform the design of a new combination technique in Section 4.3.

Methodology for Selecting Papers: I started surveying papers in Google Scholar using keywords (e.g., "combining <defense 1> and <defense 2>"). I selected all papers including those published in top-tier ML and security/privacy venues (e.g., NeurIPS, ICML, ICLR, AAAI, CCS, S&P), related workshop papers, and unpublished papers on ArXiv. I examined their citations and related work to find other papers. Finally, I used papers from related surveys (e.g., [219, 221, 231, 50, 372]) to ensure a comprehensive coverage.

Table 4.1: **Overview of pairwise combinations among defenses**: For each combination cell, I cite related work and indicate the "interaction type" (● → alignment, ● → conflict, or ● → unexplored), and "combination technique" used (**T1** or **T2**).

	EvsnRob	PoisnRob	MdlWM	Fngrprnt	DtWM	DiffPriv	GpFair
PoisnRob	● → **T1**: [338, 339]						
MdlWM	● → **T2**: [236] ● → **T2**: [373]	●					
Fngrprnt	● → **T2**: [236] ● → **T2**: [172]	●	●				
DtWM	● → **T2**: [236]	●	●	●			
DiffPriv	● → **T1**: [340, 341, 369, 370, 342, 343]	● → **T2**: [374, 375, 376]	● → **T2**: [236]	● → **T2**: [236]	● → **T2**: [236]		
GpFair	●: **T1**: [259, 344, 377, 345, 346, 347, 348, 349, 350, 351] ● → **T2**: [348]	● → **T1**: [352, 353, 295, 354]	●	●	●	● → **T1**: [355, 356, 357, 358, 307, 359, 360, 361, 362, 306, 363, 364] ● → **T2**: [337]	
Expl	● → **T1**: [365, 366, 367]	●	●	●	●	● → **T1**: [368]; **T2**: [371]	●

Survey of Prior Work

I describe the defense combinations in the order of appearance along the columns in Table 4.1. I use the acronyms for the defenses from Chapter 2: Table 2.2 below. For related work, I indicate the nature of the interaction between the defenses, and the combination technique used, in the following format: (<nature of interaction; ●or ●>: <combination technique; **T1** or **T2**>).

EvsnRob + PoisnRob

EvsnRob suppresses adversarial examples (as outliers) while PoisnRob suppresses poisons (as outliers) in $\mathcal{D}_{tr}$. Hence, their objectives seem to be aligned as shown by Hu et al. [339] who use a bi-level optimization to protect against both poisons and adversarial examples (●: **T1**). Additionally, by viewing PoisnRob as out-of-distribution (OOD) generalization, one can modify adversarial training to incorporate adversarial examples from the new domain to improve domain generalization [338]. This allows the model to learn robust features for OOD generalization, thereby aligning with PoisnRob (●: **T1**).

EvsnRob + MdlWM

Adversarial training (EvsnRob-In) suppresses the influence of backdoors which are used for MdlWM-Pre (●: **T2**) [236]. However, generating adversarial-example based watermarks with a higher ϵ_{rob} than EvsnRob-In can result in an effective combination (●: **T2**) [373]. Hence, instead of standard backdoor-based watermarks, optimizing the watermarks to avoid conflict can result in different interactions between MdlWM-Pre with EvsnRob.

EvsnRob + DtWM

Radioactive data (as DtWM-Pre) adds backdoors as watermarks to $\mathcal{D}_{tr}$ by perturbing the inputs (similar to adversarial examples) [110]. Hence, adversarial training will suppress the influence of watermarks used for DtWM (●: **T2**) [236].

EvsnRob + Fngrprnt

Dataset inference (Fngrprnt) [174] is effective with EvsnRob and incurs an acceptable performance drop (●: **T2**) [236]. I attribute this to the defenses being applied at different stages (in-training vs. post-training), which reduces conflict between them. On the other hand, a variant of Fngrprnt based on adversarial examples (i.e., "conferrable examples") is ineffective when EvsnRob is applied to either $\mathcal{M}$ or $\mathcal{M}^{sur}$ (●: **T2**) [172]. I mark these two cases with opposing nature of interactions separately in Table 4.1.

EvsnRob + DiffPriv

Hayes et al. [378] show that the generalization is worse on combining the objectives of adversarial training (EvsnRob-In) and DPSGD (DiffPriv-In), suggesting a conflict (●: **T1**). Bu et al. [340] modify the minimax objective function of adversarial training to include DPSGD without violating its guarantees (●: **T1**). Wu et al. [341] combine randomized smoothing with DPSGD by averaging the gradients over multiple augmentations of data records in $\mathcal{D}_{tr}$ before clipping to account for the privacy budget of adversarial examples. Both techniques modify the objective function (●: **T1**).

Training $\mathcal{M}$ on public data, combined with appropriate hyperparameter choices for DiffPriv and followed by task-specific fine-tuning, can result in a better trade-off (●: **T1**) [342, 343]. Other works have explored alternative optimizations, such as adding DiffPriv noise to both

input and hidden layers, using ensemble adversarial training to incorporate adversarial examples into private $\mathcal{D}_{tr}$, and modifying the objective function to provide DiffPriv guarantees for adversarial examples (●: **T1**) [370, 369].

EvsnRob + GpFair

Adversarial training (EvsnRob-In) and group fairness (GpFair-In) have conflicting objectives: training $\mathcal{M}$ with EvsnRob-In pushes the decision boundary away from $\mathcal{D}_{tr}$, while GpFair-In brings it closer [259]. Also, EvsnRob-In increases the disparity among demographic subgroups due to class imbalance in $\mathcal{D}_{tr}$ [237], and long-tailed distribution [350, 61, 346, 237]. Several works modify EvsnRob-In's objective function to ensure equitable model behavior across demographic subgroups by assigning higher weight to minority subgroup (●: **T1**) [344, 61, 345, 346, 347, 348, 350, 349]. Wei et al. [351] use different training configurations and assigning different weights to different classes to improve class-wise robustness (●: **T1**).

EvsnRob + Expl

Adversarial training (EvsnRob-In) improves the interpretability of the gradients [138]. This suggests an alignment with Expl [379]. Also, both defenses can be combined using a minimax objective to construct high fidelity Expl, and resist adversarial examples (●: **T1**) [365, 366, 367].

PoisnRob + DiffPriv

DiffPriv reduces the influence of outliers thereby improving robustness against poisons as shown in several works [374, 375, 300, 299]. Hence, DPSGD mitigates poisons and the defenses are aligned, resulting in an effective combination (●: **T2**).

PoisnRob + GpFair

PoisnRob may overly flag data records from the minority groups as outliers for removal, which increases the bias [295]. This can be corrected by reweighing the scores assigned to outliers to account for sensitive attributes (●: **T1**) [352, 353]. Also, an outlier detector can be trained to minimize the correlation between outlier scores and sensitive attributes, thus aligning PoisnRob with GpFair (●: **T1**) [295, 354].

MdlWM + DiffPriv

DPSGD (DiffPriv-In) reduces memorization of data records in $\mathcal{D}_{tr}$. Since backdoor-based watermarks are typically outliers, DiffPriv-In makes watermarks ineffective (MdlWM). Hence, MdlWM conflicts with DiffPriv (●: T2) [236].

DtWM + DiffPriv

Ideally, DPSGD (DiffPriv-In) suppresses watermarks (DtWM-Pre), suggesting a conflict. However, empirical results suggest that DtWM-Pre was effective when combined with DPSGD [236]. A possible explanation is that the adversarial example-based watermarks [110], are inliers and are not suppressed by DPSGD (●: T2) [236].

Fngrprnt + DiffPriv

Szyller and Asokan [236] found that DPSGD (DiffPriv-In) and dataset inference (Fngrprnt) do not conflict, though no reason was provided. I attribute this to defenses being applied in different stages in the ML pipeline, thus, reducing conflict (●: T2).

DiffPriv + GpFair

DPSGD (DiffPriv-In) shows disparate behavior over demographic subgroups [251]. Theoretically, it is impossible to design a high utility binary classifier that is private and fair [380, 381]. Several works modify the objective function by using fairness constraints, regularization, and game theoretic optimization (●: T1) [355, 356, 357, 307, 358, 359, 382]. Yaghini et al. [337] combine demographic parity regularization with DPSGD, and estimate fairness on a public dataset to avoid consuming extra privacy budget (●: T1). Also, functional mechanism adds Laplace noise to the objective function, along with variable noise for different subgroups, which reduces discrimination (●: T1) [360, 361]. However, this is limited to the convex objective functions (e.g., logistic regression). Esipova et al. [306] attribute unfairness in DPSGD to the differences in unclipped and clipped gradient directions. Subsequently, several works have proposed variable gradient clipping to minimize discriminatory behavior while maintaining utility (●: T1) [363, 364, 362]. Yaghini et al. [337] use PATE framework to apply fairness constraints and DiffPriv noise in the aggregated votes from the teacher's ensemble. Both fairness and privacy are applied in pre-training (●: T2).

`DiffPriv` + `Expl`

The objectives of both these defenses are inherently conflicting: `DiffPriv` hides information to minimize leakage while `Expl` releases additional information to improve comprehensibility [383]. Yang et al. [368] train an autoencoder with `DiffPriv` (functional mechanism). This autoencoder is used to generate data records and compute counterfactual data records that satisfy `DiffPriv` via the post-processing property (●: **T1**). Patel et al. [371] propose an adaptive DPSGD algorithm to generate high-quality explanations without consuming ϵ_{dp}, by reusing past explanations for similar data records (●: **T2**).

4.2 Insights from Systematization

Based on the systematization, I identify and discuss unexplored combinations, requirements for an ideal combination technique, and the limitations of prior techniques.

Unexplored Combinations

From Table 4.1, I identify 14 unexplored combinations which are marked as ●:

- `PoisnRob` with {`MdlWM`, `DtWM`, `Fngrprnt`, `Expl`}
- `MdlWM` with {`DtWM`, `Fngrprnt`, `GpFair`, `Expl`}
- `DtWM` with {`Fngrprnt`,`GpFair`,`Expl`}
- `Fngrprnt` with {`GpFair`, `Expl`}
- `GpFair` with `Expl`.

Takeaway: Unexplored combinations reveal research gaps which allow identifying novel conflicting and non-conflicting defense combinations. This also highlights opportunities for future research to design novel combination techniques.

I revisit these unexplored combinations in the evaluation in Section 4.4 and 4.5.

Evaluating Combination Techniques

A combination technique should allow a practitioner to quickly determine whether defenses can be effectively combined. Empirical evaluation to determine the effectiveness of a combination, while definitive, can be expensive, especially when multiple defenses are involved. I outline requirements for an ideal technique, and identify limitations of prior techniques.

Requirements: An ideal combination technique should be:

R1 *Accurate:* correctly identifies whether a combination is effective or not
R2 *Scalable:* allows two or more defenses to be combined simultaneously
R3 *Non-invasive:* does not require modifying defenses allowing easy adoption
R4 *General:* applicable to various defenses.

Combination techniques should be evaluated for the four requirements (R1-R4).

Limitations of Prior Techniques: I summarize the limitations of existing techniques (T1-T2) as per the requirements (R1-R4) in Table 4.2. I use ○ when a requirement is not satisfied, ⊛ for partially satisfied, and ● for fully satisfied.

T1 **Optimization-based technique** where modifying the objective function for training, followed by hyperparameter tuning can result in an effective combination. However, this often results in a trade-off between effectiveness of constituent defenses and model utility. Hence, I mark T1 as partially accurate (R1 → ⊛). This trade-off also explains why prior work has struggled to scale beyond two defenses (R2 → ○). Furthermore, some defenses are not applicable for T1 (e.g., `Expl` and `Fngrprnt`), and require modifications or non-standard variants (R3 → ○). Finally, these optimizations are specific to the defenses being combined and model types (e.g., logistic regression). These do not generalize to other defense combinations or other models (e.g., neural networks), thus, have limited applicability (R4 → ○).

Table 4.2: **Requirements satisfied by the combination techniques**: ○ → Not satisfied; ⊛ → Partially satisfied; ● → Fully satisfied.

Technique	R1 (Accurate)	R2 (Scalable)	R3 (Non-Invasive)	R4 (General)
T1	⊛	○	○	○
T2	⊛	●	●	●

T2 **Mutually Exclusive Placement** suggests that defenses applied at the same stage of the ML pipeline are likely to conflict, whereas those applied at different stages are unlikely to do so. Here, we can apply up to one defense in each of the three stages, thus, making it scalable (R2 → ●). Defenses do not need any modification (R3 → ●), and the combination technique is applicable to all types of defenses (R4 → ●). However, the combinations may not be effective: (i) a defense in a later stage of the pipeline can conflict with earlier ones (false negatives) [236], and (ii) it rules out defenses in the same stage that do not conflict (false positives, see Section 4.5). Hence, there are errors while identifying effective combinations (partially accurate R1 → ⊛).

Takeaway: *Neither technique satisfies all the requirements.* T2 is promising as it satisfies R2, R3, and R4, but not R1 (incurs false positives and false negatives).

4.3 Design of Def\Con (Proposed Combination Technique)

While T2 is promising, it overlooks underlying factors for conflicts, leading to false positives/negatives. *Can we address the limitations of T2 and improve R1?* Recall from Section 4.2 that conflicts arise when (i) an early-stage defense uses a risk which is mitigated by a later-stage defense, or (ii) changes by an early-stage defense are overridden by a later-stage defense. I conjecture that by accounting for these underlying factors, R1 can be met. I present Def\Con, a principled technique to identify whether defenses can be effectively combined by accounting for the factors underlying conflicting interactions among defenses.

Methodology to Derive Def\Con: I start with the naïve technique and modify it to take the underlying factors influencing conflicting interactions among defenses into account. I iterate over the design of Def\Con using prior work in Section 4.1, and evaluate on unexplored combinations as shown later in Section 4.5.

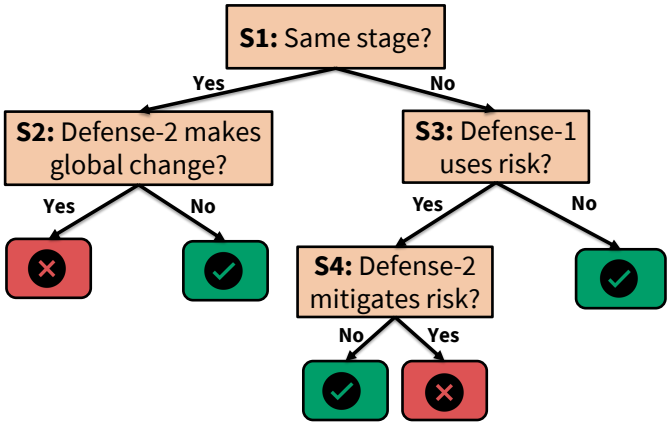


Figure 4.1: **Flowchart depicting various steps in Def\Con** to identify a conflict between $\texttt{Def}_1$ and $\texttt{Def}_2$ ($\texttt{Def}_2$ is applied after $\texttt{Def}_1$).

Def\Con Description: I describe Def\Con using the example of combining two defenses, $\texttt{Def}_1$ and $\texttt{Def}_2$ which protect against $\texttt{Rk}_1$ and $\texttt{Rk}_2$ respectively, and later discuss how to extend to more than two defenses. I start by identifying variants of each of the defenses across pre-, in-, and post-training stages (see Table 2.2). I compare each variant of $\texttt{Def}_1$ with that of $\texttt{Def}_2$, and use $\triangle$ for alignment, and Δ for conflict. Assuming $\texttt{Def}_1$ is applied first and then $\texttt{Def}_2$, we follow the steps below *in sequence*:

S1 Are $\texttt{Def}_1$ and $\texttt{Def}_2$ applied in the same stage?
- If yes, go to S2

- Else, go to **S3**

S2 The type of changes made by the defenses determines whether there is a conflict. I classify the changes as *global*, *local*, and *none*. *Global changes* modify $\mathcal{M}$ (e.g., training with a regularization term, pruning) or transform all records in $\mathcal{D}_{tr}$ (e.g., synthetic data generation for DP or fairness during pre-training). *Local changes* affect specific data records (e.g., adding watermarks in pre-training or modifying certain predictions in post-training). `Fngrprnt-Post` and `Expl-Post` make *no changes* to $\mathcal{M}$ and $\mathcal{D}_{tr}$.

- If $\texttt{Def}_1$ makes global/local/no changes while $\texttt{Def}_2$ makes local/no changes, I mark this as Δ.
  Rationale: *Changes by $\texttt{Def}_1$ will not interfere with local/no changes by $\texttt{Def}_2$, as $\texttt{Def}_1$ is applied first. Hence, no conflict.*
- If $\texttt{Def}_1$ makes global/local/no changes while $\texttt{Def}_2$ makes global changes, I mark as Δ.
  Rationale: *Global changes by $\texttt{Def}_2$ will override changes by $\texttt{Def}_1$, thereby reducing its effectiveness. This is called catastrophic forgetting when the defenses are applied sequentially during training [384, 236]. This is a conflict.*

S3 $\texttt{Def}_1$ and $\texttt{Def}_2$ are in different stages. Does $\texttt{Def}_1$ use a risk `Rk` as part of the defense (e.g., watermarking uses backdoors)?

- If yes, go to **S4**.
- If no, mark as Δ.
  Rationale: *If $\texttt{Def}_1$ does not use `Rk`, the susceptibility to `Rk` will not be impacted after applying $\texttt{Def}_2$. Hence, $\texttt{Def}_1$ and $\texttt{Def}_2$ are unlikely to interfere with each other.*

S4 Does $\texttt{Def}_2$ protect against `Rk` either explicitly or via unintended interaction?

- If yes, mark as Δ.
  Rationale: *Since $\texttt{Def}_1$ uses `Rk` (either explicitly or via unintended interaction), $\texttt{Def}_2$ will reduce susceptibility to `Rk` making $\texttt{Def}_1$ less effective. Hence, there is a conflict.*
- If no, mark as Δ.
  Rationale: *$\texttt{Def}_1$ and $\texttt{Def}_2$ are unlikely to interfere with each other. Hence, there is no conflict.*

I summarize the steps in `Def\Con` in Figure 4.1. `Def\Con` evaluates combination effectiveness based solely on the effectiveness of the constituent defenses, without considering the model utility. I discuss model utility in Section 4.6.

Note on `DiffPriv`: Combining `DiffPriv` with other defenses does not consume additional privacy budget, as I assume that any modifications to $\mathcal{D}_{tr}$ (e.g., adding watermarks) are performed within the privacy boundary[2]. Any defense applied after `DiffPriv` (e.g., modifying $\mathcal{M}$ or applying filters to inputs/outputs) is "free" due to the post-processing property.

[2]This is why, in the evaluation (Table 4.6), we do not report changes in ϵ_{dp} after combining defenses. If, however, these combinations fall outside the privacy boundary, additional analysis is required to estimate the resulting privacy budget. I leave this as future work (Chapter 9: Section 9.3.1) and follow the common assumption that all operations occur within the privacy boundary.

I discuss how to extend Def\Con to cover more than two defenses in Section 4.6. I will now comprehensively evaluate Def\Con with respect to the requirements (**R1**-**R4**).

4.4 Experimental Setup

I present the datasets and metrics used for evaluation (Section 4.4.1), the choice of defenses (Section 4.4.2), and their threat models (Section 4.4.3), and finally, the evaluation metrics and implementation details (Section 4.4.4).

4.4.1 Datasets and Models

I use two image datasets: FMNIST and UTKFACE. FMNIST consists of 28x28 grayscale images of ten clothing types, with 60,000 train and 10,000 test data records. I classify these using a two layer CNN with 16 and 32 filters, ReLU activation, and a fully connected layer for ten-class classification. UTKFACE includes 48x48 RGB images, classifying individuals as young (under 30), with 11,852 train and 10,667 test data records. It also includes the sex of the individuals as a sensitive attribute. I use a VGG16 model with a fully connected layer for binary classification. I select FMNIST since all of the defenses I consider for evaluation (in Section 4.4.2) have used it in their experiments. I select UTKFACE because many defenses effective for FMNIST are likely to be applicable to it as well, given that both are image datasets. Also, UTKFACE includes sensitive attributes, making it suitable for GpFair.

4.4.2 Choosing Defenses for Evaluation

To select defenses for the evaluation, I start with the ones described in Chapter 2: Section 2.4 (Table 2.2). I remove defenses that are not robust: EvsnRob-Post (Input Processing) and PoisnRob-Pre (Data Sanitization) [385, 386]. I exclude those that were ineffective on the considered datasets: EvsnRob-Pre (Data Augmentation) [131, 132, 133], DiffPriv-Pre (Private Data) [181], GpFair-Pre (Fair Data) [190], and GpFair-Post (Calibration) [199]. I speculate about these defenses later in Chapter 9: Section 9.3. We are left with eleven defenses:

- EvsnRob-In (adversarial training)
- PoisnRob-In (fine-tuning)
- PoisnRob-Post (model pruning)
- MdlWM-Pre (backdoor watermarks)

- `MdlWM-In` (watermarks via objective function)
- `MdlWM-Post` (API-based watermarks)
- `DtWM-Pre` (backdoor watermarks)
- `Fngrprnt-Post` (dataset inference)
- `DiffPriv-In` (DPSGD)
- `GpFair-In` (regularization)
- `Expl-Post` (attribution).

There are 55 pairwise combinations from the above defenses. I exclude combinations among defenses with the same objective which includes three combinations among watermarking defenses (`MdlWM-Pre`, `MdlWM-In`, `MdlWM-Post`), three combinations of `Fngrprnt-Post` with `MdlWM-Pre`, `MdlWM-In`, `MdlWM-Post`, and finally, the combination between `PoisnRob-In` and `PoisnRob-Post`. This leaves us with 48 combinations. I also remove ten combinations where both defenses are applied during in-training. Here, both `Def\Con` and the naïve technique predict a conflict (i.e., Δ and Ψ respectively). To apply both defenses in the in-training phase, they can be modified for an effective combination (marked as **T1** in Section 4.2). However, this makes it invasive (violates **R3**). Also, defenses can be combined sequentially (e.g., pre-training on the first defense, then fine-tuning on the second), or by alternating the training of both defenses every few epochs. Since, these are non-standard approaches to apply existing defenses, I omit the comprehensive evaluation of these ten combinations.Finally, ***we are left with 38 combinations for evaluation.***

4.4.3 Threat Models for Defenses

For each of the eleven defenses selected in Section 4.4.2, I adopt the same threat model defined in their respective works. I describe the threat models using the notation introduced in Chapter 2: Section 2.3.

- **`EvsnRob-In`**: $\mathcal{A}dv$ is a malicious client actively inducing misclassification by manipulating inputs ($\overline{\texttt{obj}} : \{\texttt{Evsn}\}$, $\overline{\texttt{role}} : \{\texttt{Clnt}\}$, $\overline{\texttt{cap}} : \{\texttt{Mal}\}$). $\mathcal{A}dv$ has white-box access to $\mathcal{M}$ and obtains the full prediction vector as output ($\overline{\texttt{k}_\texttt{m}} : \{\texttt{WBox.fullPred}\}$), allowing $\mathcal{A}dv$ to compute worst-case adversarial examples using feedback from $\mathcal{M}$ ($\overline{\texttt{opt}} : \{\texttt{Adpt}\}$). $\mathcal{A}dv$ does not have access to $\mathcal{D}_{tr}$, but instead uses a non-overlapping dataset to generate adversarial examples with access to classification labels ($\overline{\texttt{k}_\texttt{d}} : \{\texttt{NoOvlap.NoBlind}\}$). There is no limit on the number of queries to $\mathcal{M}$ ($\overline{\texttt{i}} : \{\texttt{Unlmtd}\}$).

$$\texttt{TM}^{\texttt{Evsn}} = \left(\begin{array}{l} \overline{\texttt{obj}} : \{\texttt{Evsn}\},\ \overline{\texttt{role}} : \{\texttt{Clnt}\},\ \overline{\texttt{cap}} : \{\texttt{Mal}\}, \\ \overline{\texttt{opt}} : \{\texttt{Adpt}\},\ \overline{\texttt{k}_\texttt{m}} : \{\texttt{WBox.fullPred}\}, \\ \overline{\texttt{i}} : \{\texttt{Unlmtd}\},\ \overline{\texttt{k}_\texttt{d}} : \{\texttt{NoOvlap.NoBlind}\} \end{array} \right)$$

- **PoisnRob-In and PoisnRob-Post:** $\mathcal{A}dv$ is a malicious data owner (DtOwnr) or data provider (DtProv) who injects poisoned or backdoored data records into $\mathcal{D}_{tr}$ to induce misclassifications ($\overline{\texttt{obj}} : \{\texttt{Poisn}\}$, $\overline{\texttt{role}} : \{\texttt{DtOwnr/DtProv}\}$). $\mathcal{A}dv$ actively deviates from the intended behavior and has partial overlap with $\mathcal{D}_{tr}$, with access to classification labels ($\overline{\texttt{cap}} : \{\texttt{Mal}\}$, $\overline{\texttt{k}_\texttt{d}} : \{\texttt{PartOvlap.NoBlind}\}$). $\mathcal{A}dv$ does not have access to $\mathcal{M}$ to optimize poisons or backdoors ($\overline{\texttt{opt}} : \{\texttt{nonAdpt}\}$, $\overline{\texttt{k}_\texttt{m}} : \{\texttt{NoAccess}\}$, $\overline{\texttt{i}} : \{\texttt{NoAccess}\}$).

$$\texttt{TM}^{\texttt{Poisn}} = \left(\begin{array}{l} \overline{\texttt{obj}} : \{\texttt{Poisn}\},\ \overline{\texttt{role}} : \{\texttt{DtOwnr}, \texttt{DtProv}\},\ \overline{\texttt{cap}} : \{\texttt{Mal}\}, \\ \overline{\texttt{opt}} : \{\texttt{nonAdpt}\},\ \overline{\texttt{k}_\texttt{m}} : \{\texttt{NoAccess}\}, \\ \overline{\texttt{i}} : \{\texttt{NoAccess}\},\ \overline{\texttt{k}_\texttt{d}} : \{\texttt{PartOvlap.NoBlind}\} \end{array} \right)$$

- **MdlWM-Pre, MdlWM-In, Fngrprnt-Post, and DtWM-Pre:** In all these defenses, $\mathcal{A}dv$ is a client attempting to steal the functionality of $\mathcal{M}$ to train $\mathcal{M}^{sur}$ ($\overline{\texttt{role}} : \{\texttt{Clnt}\}$, $\overline{\texttt{obj}} : \{\texttt{ModExt}\}$). $\mathcal{A}dv$ is passive and does not use $\mathcal{M}$'s feedback for optimizing subsequent attack queries ($\overline{\texttt{cap}} : \{\texttt{HbC}\}$, $\overline{\texttt{opt}} : \{\texttt{nonAdpt}\}$). $\mathcal{A}dv$ uses a non-overlapping $\mathcal{D}^{aux}$ to train and evaluate $\mathcal{M}^{sur}$ ($\overline{\texttt{k}_\texttt{d}} : \{\texttt{NoOvlap.NoBlind}\}$). $\mathcal{A}dv$ has blackbox access to $\mathcal{M}$ that returns the full prediction vector ($\overline{\texttt{k}_\texttt{m}} : \{\texttt{BBox.fullPred}\}$), and sends k queries same as the size of $\mathcal{D}^{aux}$ ($\overline{\texttt{i}} : \{\texttt{KShot}\}$).

$$\texttt{TM}^{\texttt{ModExt}} = \left(\begin{array}{l} \overline{\texttt{obj}} : \{\texttt{ModExt}\},\ \overline{\texttt{role}} : \{\texttt{Clnt}\},\ \overline{\texttt{cap}} : \{\texttt{HbC}\}, \\ \overline{\texttt{opt}} : \{\texttt{nonAdpt}\},\ \overline{\texttt{k}_\texttt{m}} : \{\texttt{BBox.fullPred}\}, \\ \overline{\texttt{i}} : \{\texttt{KShot}\},\ \overline{\texttt{k}_\texttt{d}} : \{\texttt{NoOvlap.NoBlind}\} \end{array} \right)$$

- **DiffPriv-In:** $\mathcal{A}dv$ is a client who infers sensitive information from $\mathcal{M}$, which outputs the full prediction vector, via inference attacks such as MemInf, AttInf, DistInf, and DtRecon ($\overline{\texttt{role}} : \{\texttt{Clnt}\}$, $\overline{\texttt{obj}} : \{\texttt{MemInf/AttInf/DistInf/DtRecon}\}$). $\mathcal{A}dv$ is passive and optimizes the attacks without access to $\mathcal{M}$ ($\overline{\texttt{cap}} : \{\texttt{HbC}\}$, $\overline{\texttt{opt}} : \{\texttt{nonAdpt}\}$). DPSGD assumes a worst-case $\mathcal{A}dv$ with access to $\mathcal{D}_{tr}$ except one data record ($\overline{\texttt{k}_\texttt{d}} : \{\texttt{PartOvlap.NoBlind}\}$). This provides a strong upper bound on privacy risks where $\mathcal{A}dv$ can have blackbox or whitebox access to $\mathcal{M}$, and sends unlimited queries (e.g., augmentations of data records) to increase attack effectiveness ($\overline{\texttt{k}_\texttt{m}} : \{\texttt{BBox/WBox.fullPred}\}$, $\overline{\texttt{i}} : \{\texttt{Unlmtd}\}$).

$$\texttt{TM}^{\texttt{MemInf/AttInf/DistInf/DtRecon}} = \left(\begin{array}{l} \overline{\texttt{obj}} : \{\texttt{MemInf/AttInf/DistInf/DtRecon}\}, \\ \overline{\texttt{role}} : \{\texttt{Clnt}\},\ \overline{\texttt{cap}} : \{\texttt{HbC}\}, \\ \overline{\texttt{opt}} : \{\texttt{nonAdpt}\},\ \overline{\texttt{k}_\texttt{m}} : \{\texttt{BBox.fullPred/WBox}\}, \\ \overline{\texttt{i}} : \{\texttt{Unlmtd}\},\ \overline{\texttt{k}_\texttt{d}} : \{\texttt{PartOvlap.NoBlind}\} \end{array} \right)$$

- **GpFair-In and Expl-Post**: Unlike other risks with explicit assumptions about $\mathcal{A}dv$, Discr does not assume an adversary. For uniformity, I treat Discr as if it were measured by $\mathcal{A}dv$.

Here, $\mathcal{A}dv$ is a client ($\overline{\texttt{obj}} : \{\texttt{Discr}\}$, $\overline{\texttt{role}} : \{\texttt{Clnt}\}, \overline{\texttt{cap}} : \{\texttt{HbC}\}$) with blackbox access to $\mathcal{M}$ ($\overline{\texttt{k}_\texttt{m}} : \{\texttt{BBox.fullPred}\}, \overline{\texttt{i}} : \{\texttt{Unlmtd}\}$), who evaluates Discr on a non-overlapping $\mathcal{D}^{aux}$ ($\overline{\texttt{k}_\texttt{d}} : \{\texttt{NoOvlap.NoBlind}\}$). $\mathcal{A}dv$ does not optimize the queries using $\mathcal{M}$'s feedback to estimate Discr ($\overline{\texttt{opt}} : \{\texttt{nonAdpt}\}$).

$$\texttt{TM}^{\texttt{Discr}} = \left(\begin{array}{l} \overline{\texttt{obj}} : \{\texttt{Discr}\},\ \overline{\texttt{role}} : \{\texttt{Clnt}\},\ \overline{\texttt{cap}} : \{\texttt{HbC}\}, \\ \overline{\texttt{opt}} : \{\texttt{nonAdpt}\},\ \overline{\texttt{k}_\texttt{m}} : \{\texttt{BBox.fullPred}\}, \\ \overline{\texttt{i}} : \{\texttt{Unlmtd}\},\ \overline{\texttt{k}_\texttt{d}} : \{\texttt{NoOvlap.NoBlind}\} \end{array} \right)$$

After combining the defenses, I evaluate the effectiveness of each constituent defense using its original evaluation metrics and threat model, as if the defenses were applied independently.

4.4.4 Metrics and Implementations

I describe the metrics for evaluating the effectiveness of each defense, and the implementations taken from publicly available code from prior work. I measure ϕ_{acc} on $\mathcal{D}_{te}$ for all defenses. The implementations for defenses are based on the state-of-the-art (Fngrprnt, PoisnRob), standard libraries (DiffPriv, GpFair, Expl), or seminal work (EvsnRob, DtWM, MdlWM). I use the standard hyperparameters which are either from the literature or the library documentation, such that the resulting individual defenses are effective (Table 4.3). When combining defenses, I use the same hyperparameters, but I revisit hyperparameter tuning for defenses in combination to check whether it can remove conflicts (see Section 4.5.4). I report the mean and standard deviation across five runs.

Evasion Robustness (EvsnRob-In)

I use the accuracy on $\mathcal{D}^{rob}_{te}$ which is obtained by generating adversarial examples for the data records in $\mathcal{D}_{te}$:

$$\phi_{\text{robacc}}(\mathcal{M}, \mathcal{D}^{rob}_{te}) = \frac{1}{|\mathcal{D}^{rob}_{te}|} \sum_{(x,y)\in\mathcal{D}^{rob}_{te}} \mathbb{I}\left\{\hat{\mathcal{M}}(x) = y\right\}$$

Ideally, ϕ_{robacc} should be close to ϕ_{acc}. I use the original implementation of TRADES [135] and an implementation of AutoAttack [387] in SecML library [388] from Pintor et al. [389]. As the effectiveness of EvsnRob-In is evaluated using attacks, poorly optimized attacks can falsely suggest defense effectiveness [390, 391, 126]. I individually optimize these attacks for evaluation with defenses and their combinations, following Pintor et al. [389] to address failures

identified by various indicators (e.g., poor optimization). I evaluate across various AutoAttack variants by (a) modifying the loss function: cross entropy (CE) and difference of logits ratio (DLR), (b) applying expectation over transformations (EoT), and (c) using random starts. For FMNIST, I use DLR, CE, DLR+EoT, and CE+EoT. For UTKFACE, I use CE and CE+EoT, as DLR is not applicable for binary classification. I report the best attack (least ϕ_{robacc}).

Outlier Removal (`PoisnRob`)

I compute accuracy on $\mathcal{D}_{te}^{\text{bd}}$ which is obtained by adding backdoors to data records in $\mathcal{D}_{te}$. This accuracy is given by:

$$\phi_{\text{ASR}}(\mathcal{M}, \mathcal{D}_{te}^{\text{bd}}) = \frac{1}{|\mathcal{D}_{te}^{\text{bd}}|} \sum_{(x,y)\in\mathcal{D}_{te}^{\text{bd}}} \mathbb{I}\left\{\hat{\mathcal{M}}(x) = y\right\}$$

where y_t is the target label chosen by $\mathcal{A}dv$. Ideally, ϕ_{ASR} should be zero. I use BadNets [392] to generate poisons by adding a white patch of size 5x5 to the images, applied to 10% of $\mathcal{D}_{tr}$. For `PoisnRob-In` (Fine-tuning), I fine-tune the last layers of $\mathcal{M}$ using random sample of 10% of $\mathcal{D}_{tr}$ without poisons [393]. For `PoisnRob-Post` (Pruning), I use the implementation from Zheng et al. [161]. I sweep pruning thresholds from 0.6 to 1.5, in increments of 0.05, to get a model with highest ϕ_{acc} and lowest ϕ_{ASR}.

Model-Watermarking (`MdlWM`)

I use the accuracy on $\mathcal{D}_{te}^{\text{wmM}}$ which is obtained by adding watermarks to data records in $\mathcal{D}_{te}$. I compute this *watermark accuracy* as:

$$\phi_{\text{wmacc}}(\mathcal{M}, \mathcal{D}_{te}^{\text{wmM}}) = \frac{1}{|\mathcal{D}_{te}^{\text{wmM}}|} \sum_{(x,y)\in\mathcal{D}_{te}^{\text{wmM}}} \mathbb{I}\left\{\hat{\mathcal{M}}(x) = y\right\}$$

where y_m represents the target labels for watermarked records. Ideally, ϕ_{wmacc} should be 100% if the model is successfully watermarked. For `MdlWM-Pre` (Backdoor), I use BadNets [392], similar to Szyller and Asokan [236], by adding a white patch of size 5x5 to 10% of the images in $\mathcal{D}_{tr}$. For `MdlWM-In` (Modifying Loss), I use the certified neural network watermarking implementation by Bansal et al. [168]. For `MdlWM-Post` (API), I use DAWN [169], which flips a fraction of $\mathcal{M}$'s predictions to serve as watermarks for training $\mathcal{M}^{sur}$. Following the original work [169], the watermark is applied to 0.2% of the predictions. Unlike other watermarking schemes, I compute ϕ_{wmacc} on $\mathcal{M}^{sur}$ and not $\mathcal{M}$.

Fingerprinting (`Fngrprnt-Post`)

I use dataset inference as the fingerprinting defense which extracts feature embeddings from $\mathcal{M}$, and trains a classifier to distinguish between $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ [174]. A model is stolen if the distance of its embeddings is similar to $\mathcal{M}$ with high confidence, and verification is successful if the p-value < 0.05. I use ϕ_{pval} as the metric following Szyller and Asokan [236]. I use the step size of 1.0 for L_1 attack, 0.01 for L_2 attack, and 0.001 for L_{inf}, and 50 data records for computing p-value from the confidence regressor model.

Data-Watermarking (`DtWM-Pre`)

To determine if a dataset was used to train a model, I compare the posterior probability of 100 watermarked test data records against 100 benign ones using a pairwise t-test [394]. I calculate the rate of successful detection (ϕ_{rsd}), which reflects the percentage of correctly identified watermarked data records from $\mathcal{D}_{te}^{\text{wmD}}$ (obtained by adding watermarks to $\mathcal{D}_{te}$). Watermarks are generated using BadNets [392] where 10% of $\mathcal{D}_{tr}$ is watermarked, and I use verification code from Li et al. [394] to compute ϕ_{rsd}. Ideally, ϕ_{rsd} should be 100% for watermarked models.

Differential Privacy (`DiffPriv-In`)

I use $\phi_{\text{dp}} = \epsilon_{\text{dp}}$, following Szyller and Asokan [236], where ideally, I want a low ϵ_{dp}. I use the implementation from Opacus library [395] with a noise multiplier of 1.0 and gradient norm clipping of 1.0 as used in their tutorial for `MNIST`.

Group Fairness (`GpFair-In`)

I measure fairness using the *equalized odds gap* on $\mathcal{D}_{te}$ for sensitive attributes S and model predictions $\hat{Y}$, given by:

$$\phi_{\text{eqodds}} = P(\hat{Y} = \hat{y} | S = 0, Y = y) - P(\hat{Y} = \hat{y} | S = 1, Y = y)$$

$\forall (\hat{y}, y) \in \{0, 1\}^2$ where an ideal value of zero indicates perfect fairness. For `GpFair-In` (Regularization), I use the code from the fair fairness benchmark that adds a regularization term to penalize equalized odds violations [396]. I set the regularization hyperparameter $\lambda = 1$ which was sufficient to reduce ϕ_{eqodds} with $\sim 2\%$ drop in ϕ_{acc}.

Explanations (`Expl-Post`)

I assess the quality of explanations using *convergence delta*, which measures the error between the explanation for a data record and a baseline [397]. I report the average convergence delta across all $\mathcal{D}_{te}$ records as ϕ_{err}. I use DeepLift from Captum library which recommends using a zero vector as a baseline [334].

4.5 Evaluation

I evaluate the effectiveness of individual defenses (Section 4.5.1), compare `Def\Con` to the naïve technique (Section 4.5.2 and 4.5.3), study the impact of hyperparameter tuning on conflicts (Section 4.5.4), and confirm that `Def\Con` meets all requirements (Section 4.5.5).

4.5.1 Effectiveness of Individual Defenses

I evaluate the effectiveness of each defense by comparing with a "no defense" baseline. I report the results in Table 4.3. I also report ϕ_{acc} to provide context but do not use it to evaluate accuracy of the techniques. I find that all the defense effectiveness metrics are better than the "no defense" baseline. Once the defenses are applied, I use their respective $\phi_{(.)}$ as the "single defense" baseline to compare the effectiveness of the defense combinations later in Section 4.5.3. To evaluate ϕ_{err} for `Expl-Post`, we do not have a "no defense" baseline to compare with. Assuming the value of ϕ_{err} is "effective", I use it as the "single defense" baseline to compare the value of ϕ_{err} in combinations.

4.5.2 Accuracy against Prior Work

Among the 38 defense combinations, I identify those which have been empirically evaluated in prior work (Section 4.1 and Tables 4.5 and 4.6). This includes eight combinations (**C1**-**C8**) where the results from prior work can be used as a ground truth to compare the predictions of `Def\Con` with the naïve technique (marked as ● or ● in Tables 4.5 and 4.6). I use green and red to indicate alignment and conflict among defenses, respectively.

The prediction from a technique is accurate when Δ (or Δ) for `Def\Con`, or Ψ (or Ψ) for naïve technique, match the ground truth ● (or ●) taken from prior work in Section 4.1. Table 4.4 includes additional details to make predictions when using `Def\Con`.

Table 4.3: **Effectiveness of defenses:** For metrics, I use ↑ (resp. ↓) where a higher (or lower) value is better, and "x" is shorthand for $\phi_{(x)}$. (ϕ_{acc} is for context).

Defense	Metric	FMNIST	UTKFACE
No Defense	**acc** (↑)	90.97 ± 0.18	80.28 ± 1.26
	robacc (↑)	7.96 ± 1.24	0.00 ± 0.00
	ASR (↓)	99.95 ± 0.04	99.98 ± 0.05
	wmacc.Pre (↑)	9.98 ± 0.28	0.00 ± 0.00
	wmacc.In (↑)	6.28 ± 1.20	62.21 ± 6.03
	wmacc.Post (↑)	0.00 ± 0.00	13.33 ± 6.32
	RSD (↑)	0.00 ± 0.00	0.00 ± 0.00
	eqodds (↓)		28.10 ± 6.34
	dp (↓)	∞	∞
`EvsnRob-In`	**acc** (↑)	86.93 ± 0.23	73.38 ± 1.15
	robacc (↑)	76.59 ± 0.28	37.38 ± 1.30
`PoisnRob-In`	**acc** (↑)	89.38 ± 0.28	79.02 ± 0.30
	ASR (↓)	9.94 ± 0.24	56.62 ± 37.83
`PoisnRob-Post`	**acc** (↑)	86.48 ± 2.35	65.42 ± 3.27
	ASR (↓)	66.44 ± 21.30	8.59 ± 16.41
`MdlWM-Pre`	**acc** (↑)	90.15 ± 0.27	79.79 ± 0.39
	wmacc (↑)	99.91 ± 0.05	100.00 ± 0.00
`MdlWM-In`	**acc** (↑)	80.87 ± 0.88	66.71 ± 10.19
	wmacc (↑)	85.61 ± 2.50	93.74 ± 11.00
`MdlWM-Post`	**acc** (↑)	90.56 ± 0.34	80.82 ± 0.45
	wmacc (↑)	100.00 ± 0.00	78.10 ± 9.33
`DtWM-Pre`	**acc** (↑)	90.31 ± 0.27	79.93 ± 0.37
	RSD (↑)	100.00 ± 0.00	100.00 ± 0.00
`Fngrprnt-Post`	**acc** (↑)	No change	No change
	pval (↓)	< 0.05	< 0.05
`DiffPriv-In`	**acc** (↑)	86.82 ± 0.11	74.07 ± 0.28
	dp (↓)	ϵ_{dp} = 1.36	ϵ_{dp} = 2.89
`GpFair-In`	**acc** (↑)		76.85 ± 1.99
	eqodds (↓)		10.89 ± 2.84
`Expl-Post`	**acc** (↑)	No change	No change
	err (↓)	0.12 ± 0.03	0.59 ± 0.05

- **C1 (`GpFair-Pre` + `DiffPriv-Pre`)** can be combined effectively in the pre-training stage (●) [337]. Naïve technique predicts Ψ (same stage) while `Def\Con` predicts Δ (defenses make local changes in **S2**).
- **C2 (`EvsnRob-In` +`Fngrprnt-Post`)** can be effectively combined (●) [236]. Naïve technique predicts Ψ (different stages) while `Def\Con` predicts Δ (**S3**=no).

Table 4.4: **Decision points for `Def\Con`:** For each defense, I identify key parameters used in evaluating combination effectiveness (Figure 4.1): type of change in **S2** (Global, Local, or None); whether the defense uses a risk in **S3** ("Yes" for backdoors or adversarial examples); and whether the defense protects against a risk in **S4**.

Defense	S2	S3	S4
`EvsnRob-In`	Global	No	Yes
`PoisnRob-In`	Global	No	Yes
`PoisnRob-Post`	Global	No	Yes
`MdlWM-Pre`	Local	Yes	No
`MdlWM-In`	Global	Yes	No
`MdlWM-Post`	Local	No	No
`DtWM-Pre`	Local	Yes	No
`Fngrprnt-Post`	None	No	No
`DiffPriv-In`	Global	No	Yes
`GpFair-In`	Global	No	No
`Expl-Post`	None	No	No

- **C3 (`DiffPriv-In` + `Fngrprnt-Post`)** can be effectively combined (🟢) [236]. Naïve technique predicts Ψ (different stages) while `Def\Con` predicts Δ (**S3**=no).
- **C4 (`MdlWM-Pre` +`EvsnRob-In`)** cannot be effectively combined (🔴) [236]. Naïve technique predicts Ψ (different stages) while `Def\Con` predicts Δ (`EvsnRob-In` makes poisons ineffective via unintended interaction in **S4**).
- **C5 (`DtWM-Pre` +`EvsnRob-In`)** cannot be effectively combined (🔴) [236]. Naïve technique predicts Ψ (different stages) while `Def\Con` predicts Δ (`EvsnRob-In` makes poisons ineffective via unintended interaction in **S4**).
- **C6 (`MdlWM-Pre` + `DiffPriv-In`)** cannot be effectively combined (🔴) [236]. Similar to **C5**, `Def\Con` predicts this as Δ while the naïve technique predicts Ψ.
- **C7 (`DtWM-Pre` + `DiffPriv-In`)** can be effectively combined (🟢) [236]. Naïve technique predicts Ψ (different stages) while `Def\Con` predicts Δ (`DiffPriv-In` reduces effectiveness of poisons via unintended interaction in **S4**). Unlike backdoor-based watermarks used in this work, adversarial example-based watermarks used by Szyller and Asokan [236], are inliers which are not suppressed by `DiffPriv-In`. Hence, `Def\Con`'s prediction differs.
- **C8 (`DiffPriv-In` + `Expl-Post`)** can be effectively combined (🟢) [371]. Naïve technique predicts Ψ (different stages) and `Def\Con` predicts Δ (**S3**=no).

Of the eight combinations, `Def\Con` predicts seven correctly, while naïve technique predicts four. This gives a balanced accuracy of 90% (TP=4, TN=3, FP=0, FN=1) for

Def\Con, and 40% (TP=4, TN=0, FP=3, FN=1) for the naïve technique.

4.5.3 Accuracy via Empirical Evaluation

I now empirically evaluate the remaining, previously unexplored, combinations to obtain the ground truth and then compute the accuracy of the predictions from both techniques.

Table 4.5: **Evaluating combinations [C9-C24]:** For brevity, "x" in **Metric** column is shorthand for $\phi_{(x)}^{\hat{\texttt{Def}}}$ where $\hat{\texttt{Def}}$ is the combination. I use ↑ (resp. ↓) to indicate if a higher (lower) value is better. For defense effectiveness, I use green when $\phi_{(.)}^{\hat{\texttt{Def}}}$ is better or equal to "single defense" baseline; orange for better or equal to "single defense" but worse than "no defense"; red for worse than "no defense". For technique predictions, I use symbol Δ (resp. Ψ) to refer to Def\Con (naïve technique) and a color code green (resp. red) to indicate alignment (conflict) among defenses. $\texttt{Def}_1$ and $\texttt{Def}_2$ indicate order of applying defenses. (ϕ_{acc} is for context).

	Combinations	Metric	FMNIST	UTKFACE		Combinations	Metric	FMNIST	UTKFACE
C9	$\texttt{Def}_1$: EvsnRob-In $\texttt{Def}_2$: MdlWM-Post (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **robacc** (↓)	90.38 ± 0.22 100.00 ± 0.00 80.43±0.85	72.79 ± 0.53 80.95 ± 7.13 42.00 ± 0.49	**C17**	$\texttt{Def}_1$: GpFair-In $\texttt{Def}_2$: Fngrprnt-Post (Ψ, Δ)	**acc** (↑) **pval** (↓) **eqodds** (↓)		78.67 ± 1.46 0.68 ± 0.21 7.46 ± 5.43
C10	$\texttt{Def}_1$: PoisnRob-In $\texttt{Def}_2$: Fngrprnt-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **pval** (↓)	89.50 ± 0.21 9.94 ± 0.22 <0.05	79.25 ± 1.06 56.09 ± 12.98 <0.05	**C18**	$\texttt{Def}_1$: GpFair-In $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **err** (↓) **eqodds** (↓)		80.52 ± 0.44 0.16 ± 0.06 12.62 ± 4.20
C11	$\texttt{Def}_1$: PoisnRob-Post $\texttt{Def}_2$: Fngrprnt-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **pval** (↓)	84.73 ± 1.72 61.36 ± 23.96 <0.05	63.70 ± 3.87 0.02 ± 0.03 <0.05	**C19**	$\texttt{Def}_1$: PoisnRob-In $\texttt{Def}_2$: MdlWM-Post (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **ASR** (↓)	89.53 ± 0.36 100.00 ± 0.00 10.48 ± 0.46	79.00 ± 0.56 69.52 ± 6.46 38.90 ± 38.73
C12	$\texttt{Def}_1$: EvsnRob-In $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **err** (↓) **robacc** (↑)	87.10 ± 0.21 0.22 ± 0.01 79.00 ± 0.21	73.65 ± 1.21 0.15 ± 0.04 39.27 ± 0.68	**C20**	$\texttt{Def}_1$: MdlWM-Post $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **err** (↓)	90.93 ± 0.18 100.00 ± 0.00 0.11 ± 0.02	80.53 ± 0.23 72.38 ± 3.56 0.55 ± 0.02
C13	$\texttt{Def}_1$: GpFair-In $\texttt{Def}_2$: PoisnRob-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **eqodds** (↓)		66.73 ± 3.24 20.21 ± 39.90 2.72 ± 3.20	**C21**	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: PoisnRob-In (Ψ, Δ)	**acc** (↑) **ASR** (↓) **RSD** (↑)	89.46 ± 0.32 10.18 ± 0.40 0.00 ± 0.00	79.00 ± 0.67 77.39 ± 35.23 80.00 ± 40.00
C14	$\texttt{Def}_1$: MdlWM-Pre $\texttt{Def}_2$: GpFair-In (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **eqodds** (↓)		79.02 ± 0.40 98.88 ± 2.13 0.00 ± 0.00	**C22**	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: MdlWM-In (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **RSD** (↑)	84.45 ± 0.56 89.25 ± 3.48 100.00 ± 0.00	79.88 ± 0.27 99.98 ± 0.03 100.00 ± 0.00
C15	$\texttt{Def}_1$: GpFair-In $\texttt{Def}_2$: MdlWM-Post (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **eqodds** (↓)		76.95 ± 1.94 80.95 ± 0.00 7.87 ± 4.72	**C23**	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: PoisnRob-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **RSD** (↑)	82.90 ± 2.06 64.55 ± 21.23 80.00 ± 40.00	69.02 ± 1.96 0.01 ± 0.01 20.00 ± 40.00
C16	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: GpFair-In (Ψ, Δ)	**acc** (↑) **RSD** (↑) **eqodds** (↓)		78.97 ± 1.21 100.00 ± 0.00 0.00 ± 0.00	**C24**	$\texttt{Def}_1$: MdlWM-Pre $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **err** (↓) **wmacc** (↑)	90.18 ± 0.21 0.14 ± 0.04 99.93 ± 0.06	79.76 ± 0.63 0.02 ± 0.03 99.96 ± 0.08

Predictions from Techniques: Before evaluating 30 combinations, I denote the defenses as $\texttt{Def}_1$ and $\texttt{Def}_2$ based on the order in which they are applied. I obtain predictions from Def\Con

Table 4.6: **Evaluating combinations [C25-C38]:** The descriptions of the symbols and colors are the same as in Table 4.5.

	Combinations	Metric	FMNIST	UTKFACE		Combinations	Metric	FMNIST	UTKFACE
C25	$\texttt{Def}_1$: MdlWM-In $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **err** (↓) **wmacc** (↑)	86.94 ± 0.50 0.19 ± 0.07 98.24 ± 0.66	72.16 ± 5.13 0.37 ± 0.18 97.60 ± 3.54	**C32**	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: MdlWM-Post (Ψ, Δ)	**acc** (↑) **RSD** (↑) **wmacc** (↑)	90.24 ± 0.29 100.00 ± 0.00 100.00 ± 0.00	78.94 ± 0.95 100.00 ± 0.00 62.26 ± 3.77
C26	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **err** (↓) **RSD** (↑)	90.04 ± 0.60 0.10 ± 0.04 100.00 ± 0.00	79.03 ± 1.10 0.54 ± 0.01 100.00 ± 0.00	**C33**	$\texttt{Def}_1$: PoisnRob-Post $\texttt{Def}_2$: MdlWM-Post (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **ASR** (↓)	85.09 ± 1.94 100.00 ± 0.00 59.48 ± 24.91	67.09 ± 2.81 73.33 ± 8.83 40.20 ± 28.82
C27	$\texttt{Def}_1$: PoisnRob-In $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **err** (↓)	89.39 ± 0.24 9.79 ± 0.15 0.06 ± 0.02	78.71 ± 0.20 44.35 ± 30.07 0.47 ± 0.02	**C34**	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: MdlWM-Pre (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **RSD** (↑)	90.31 ± 0.27 99.96 ± 0.0 100.00 ± 0.00	78.53 ± 1.75 100.00 ± 0.00 100.00 ± 0.00
C28	$\texttt{Def}_1$: PoisnRob-Post $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **err** (↓)	84.62 ± 3.56 76.11 ± 15.85 0.08 ± 0.01	63.80 ± 3.37 0.00 ± 0.00 0.15 ± 0.06	**C35**	$\texttt{Def}_1$: EvsnRob-In $\texttt{Def}_2$: PoisnRob-Post (Ψ, Δ)	**acc** (↑) **robacc** (↑) **ASR** (↓)	90.39 ± 0.63 46.00 ± 1.02 79.68 ± 10.25	80.28 ± 0.39 0.00 ± 0.00 0.00 ± 0.00
C29	$\texttt{Def}_1$: Fngrprnt-Post $\texttt{Def}_2$: Expl-Post (Ψ, Δ)	**acc** (↑) **pval** (↓) **err** (↓)	90.56 ± 0.16 <0.05 0.11 ± 0.02	80.42 ± 0.59 <0.05 0.50 ± 0.03	**C36**	$\texttt{Def}_1$: MdlWM-Pre $\texttt{Def}_2$: PoisnRob-In (Ψ, Δ)	**acc** (↑) **ASR** (↓) **wmacc** (↑)	89.48 ± 0.15 10.18 ± 0.46 10.18 ± 0.46	79.20 ± 0.60 46.92 ± 36.92 46.92 ± 36.92
C30	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: Fngrprnt-Post (Ψ, Δ)	**acc** (↑) **pval** (↓) **RSD** (↑)	90.19 ± 0.59 <0.05 100.00 ± 0.00	79.80 ± 0.48 <0.05 100.00 ± 0.00	**C37**	$\texttt{Def}_1$: MdlWM-Pre $\texttt{Def}_2$: PoisnRob-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **wmacc** (↑)	82.86 ± 4.16 71.32 ± 14.11 71.31 ± 14.10	64.09 ± 3.09 0.00 ± 0.00 0.00 ± 0.00
C31	$\texttt{Def}_1$: DiffPriv-In $\texttt{Def}_2$: MdlWM-Post (Ψ, Δ)	**acc** (↑) **wmacc** (↑) **dp** (↓)	86.83 ± 0.20 100.00 ± 0.00 ϵ = 1.36	74.62 ± 0.49 79.05 ± 3.81 ϵ = 2.89	**C38**	$\texttt{Def}_1$: MdlWM-In $\texttt{Def}_2$: PoisnRob-Post (Ψ, Δ)	**acc** (↑) **ASR** (↓) **wmacc** (↑)	66.68 ± 9.80 58.59 ± 19.22 58.65 ± 19.23	73.69 ± 3.01 99.60 ± 0.37 99.73 ± 0.29

and the naïve technique, and indicate them as a tuple: (Naïve prediction, Def\Con prediction). These are indicated in Tables 4.5 and 4.6. I use the information in Table 4.4 to make predictions in S2-S4 for Def\Con.

- For defenses applied in the same stage (S1=yes), the naïve technique predicts Ψ. I get the following cases to determine the prediction from Def\Con:
  1. $\texttt{Def}_2$ makes local/no changes (S2=no), Def\Con predicts this as Ψ. I mark them as (Ψ, Δ) which include **C11**, **C20**, **C28**, **C29**, **C33**, and **C34**.
  2. $\texttt{Def}_2$ makes global changes (S2=yes), and Def\Con predicts this as Δ. I mark them as (Ψ, Δ) but I did not observe any such combinations.
- For defenses applied in different stages (S1=no), the naïve technique predicts Ψ. I have the following cases to determine the prediction from Def\Con:
  1. $\texttt{Def}_1$ does not use a risk (S3=no) and hence, $\texttt{Def}_1$ and $\texttt{Def}_2$ do not conflict. I mark them as (Ψ, Δ): **C9**, **C10**, **C12**, **C13**, **C15**, **C17-C19**, **C27**, **C31**, and **C35**.
  2. $\texttt{Def}_1$, such as MdlWM-Pre and DtWM-Pre, uses a risk (S3=yes), but $\texttt{Def}_2$ does not protect against this risk (S4=no). Hence, there is no conflict and I mark such combinations as (Ψ, Δ) which include **C14**, **C16**, **C22**, **C24-C26**, **C30**, and **C32**.

3. $\texttt{Def}_1$, such as MdlWM-Pre and DtWM-Pre, uses a risk (S3=yes), and $\texttt{Def}_2$ mitigates these risks (e.g., PoisnRob). There is a conflict and I mark them as (Ψ, Δ) which include **C21**, **C23**, **C36**-**C38**.

I evaluate the 30 combinations on FMNIST and UTKFACE (Tables 4.5 and 4.6). I compare the effectiveness metrics for each defense in the combination to the "single defense" baseline from Table 4.3. I use green to indicate that the metrics are better or similar to "single defense"; orange for worse than single defense but better than "no defense"; and red for similar or worse than "no defense". Metrics marked as orange can still be useful since it provides some protection compared to "no defense". I consider the worst-case by treating a combination as a conflict if at least one dataset has at least one metric marked as orange or red.

Of the 30 combinations, Def\Con predicts 27 correctly, while the naïve method predicts only 18. This gives a balanced accuracy of 81% (TP=22, TN=5, FP=3, and FN=0) for Def\Con compared to 36% (TP=16, TN=0, FP=8, and FN=6) for the naïve technique.

Takeaway: By explicitly accounting for the factors underlying conflicts among defenses, Def\Con achieves a higher accuracy than the naïve technique (satisfies **R1**).

4.5.4 Hyperparameter Tuning for Combinations

I now check if hyperparameter tuning can resolve conflicts to see if it can turn (i) *false positives* (predicted as aligned, but ground truth is empirically conflicting) into *true positives*, and (ii) *true negatives* (predicted and confirmed as conflict) into *false negatives*. I exclude *false negatives* (not observed in the evaluation), and *true positives*, correctly predicted and confirmed as aligned (does not require further improvement with hyperparameter tuning). I use grid search and identify various hyperparameter configurations (Table 4.7).

Do False Positives turn to True Positives?

This includes three combinations (**C17**, **C32**, **C35**):

- **C17 (GpFair-In + Fngrprnt-Post)**: This combination is conflicting as Fngrprnt-Post is ineffective ($\phi_{\text{pval}} > 0.05$), and Def\Con incorrectly predicts the combination as Δ in **S3**. I explore the following hyperparameters: regularization hyperparameter (for GpFair-In), number of iterations for optimization and the number of fingerprints (for Fngrprnt-Post). For each dataset, I have 36 experiments (= $4 \times 4 \times 3$). None of the experiments alleviated the conflict. Following prior work [236], I speculate that Fngrprnt-Post is ineffective because it relies on the decision boundary, which shifts significantly after applying GpFair-In.

Table 4.7: **Hyperparameter configurations** for tuning defenses in conflicting combinations.

Defense	Hyperparameter	Values
`EvsnRob-In`	Regularization	{2, 4, 6 (default), 8}
`PoisnRob-Post`	Pruning threshold	0.6–1.5 (step 0.05)
`MdlWM-Pre`	Trigger size Watermark fraction	{3×3, 5×5 (default)} {0.1 (default), 0.2, 0.3}
`MdlWM-In`	Watermark fraction Training noise Noise step size	{0.1, 0.2, 0.3} {0.5, 0.75, 1.0 (default), 1.25} {0.05 (default), 0.10, 0.15}
`MdlWM-Post`	Watermark fraction	{0.002 (default), 0.01, 0.02}
`DtWM-Pre`	Trigger size Watermark fraction	{3×3, 5×5 (default)} {0.1 (default), 0.2, 0.3}
`GpFair-In`	Regularization	{0.5, 1 (default), 1.5, 2}
`Fngrprnt-Post`	Iterations # Fingerprints	{25, 50 (default), 75, 100} {100 (default), 150, 200}

- **C32 (`DtWM-Pre` + `MdlWM-Post`)**: This combination is empirically non-conflicting for FMNIST but not for UTKFACE where ϕ_{wmacc} is less than the "single defense" baseline. `Def\Con` incorrectly predicts this combination as Δ in S4. I explore the following hyperparameters: (i) trigger size and watermark fraction for `DtWM-Pre`; (ii) watermark fraction for `MdlWM-Post`. For UTKFACE, I evaluate 18 experiments (=2×3×3). One experiment with 3×3 trigger size (`DtWM-Pre`), 30% watermarks (`DtWM-Pre`), and 2% watermarks (`MdlWM-Post`), I get ϕ_{acc} = 75.98 ± 0.61, ϕ_{rsd} = 100.00 ± 0.00 (green), and ϕ_{wmacc} = 70.19 ± 4.61 (green). Hence, I remove the conflict and the corresponding false positive.
- **C35 (`EvsnRob-In` + `PoisnRob-Post`)**: This combination is conflicting as `EvsnRob-In` is ineffective (poor ϕ_{robacc}), and `Def\Con` incorrectly predicts as Δ in S4. I vary the regularization hyperparameter (`EvsnRob-In`), and pruning thresholds (`PoisnRob-Post`). None of the experiments removed the conflict. I speculate that the model parameters memorizing poisons and adversarial examples overlap. Thus, pruning a model (to reduce ϕ_{ASR}) trained with `EvsnRob-In`, also reduces ϕ_{robacc}, resulting in a conflict.

Do True Negatives turn to False Negatives?

I evaluate hyperparameter tuning for five combinations (**C21**, **C23**, **C36**, **C37**, and **C38**).

- **C21 (`DtWM-Pre` + `PoisnRob-In`)**: I consider trigger size, and watermark fraction for `DtWM-Pre`. For each dataset, I get six experiments (=2×3). None of them removed the conflict since `PoisnRob-In` mitigates backdoors for `DtWM-Pre`.
- **C23 (`DtWM-Pre` + `PoisnRob-Post`)**: I tune the same hyperparameters for `DtWM-Pre` as in **C21**. For `PoisnRob-Post`, I sweep across various pruning thresholds. For each dataset, I get six experiments (=2×3). For FMNIST, trigger size of 3×3 and 30% watermarks, gives ϕ_{acc}=82.00 ± 5.50; ϕ_{rsd}= 100.00 ± 0.00; ϕ_{ASR}= 59.90 ± 12.81. This is marked as no conflict (green). However, there is a conflict for UTKFACE, making the overall combination a conflict.
- **C36 (`MdlWM-Pre` + `PoisnRob-In`)**: I tune the same hyperparameters for `MdlWM-Pre`, as `DtWM-Pre` in **C21**. For each dataset, I get six experiments (=2×3). None of them removed the conflict since `PoisnRob-In` mitigates backdoors for `MdlWM-Pre`.
- **C37 (`MdlWM-Pre` + `PoisnRob-Post`)**: I tune the same hyperparameters for `MdlWM-Pre`, as `DtWM-Pre` in **C21**. For `PoisnRob-Post`, I sweep across various pruning thresholds. For each dataset, I get six experiments (=2×3). None of them removed the conflict since `PoisnRob-Post` mitigates backdoors for `MdlWM-Pre`.
- **C38 (`MdlWM-In` + `PoisnRob-Post`)**: I tune the fraction of watermarks, training noise, and step size for `MdlWM-In`. For `PoisnRob-Post`, I sweep across various pruning thresholds. For each dataset, I get 36 experiments (=3×4×3). None of them removed the conflict since `PoisnRob-Post` mitigates the backdoors for `MdlWM-In`.

Summary: I find that hyperparameter tuning is useful for two combinations (**C23** and **C32**). For **C32**, I removed the false positive, thereby increasing `Def\Con`'s balanced accuracy to 86% (from 81%). For **C23**, I could remove conflict for one of the two datasets, but the combination was still marked as a conflict (no additional false negatives).

Takeaway: Hyperparameter tuning for *conflicting combinations* is important, to check if it turns (a) false positives to true positives, or (b) true negatives to false negatives.

Considerations for Hyperparameter Tuning Effectiveness

I identify aspects influencing the effectiveness of hyperparameter tuning and their relation to conflicting combinations:

- *Expressiveness of Hyperparameters:* Hyperparameter tuning is effective only when the relevant hyperparameters directly influence the conflicting interaction; if the interaction is not sensitive to those parameters, hyperparameter tuning cannot resolve the conflict. This may explain why hyperparameter tuning is ineffective for **C21**, **C36**, **C37**, and **C38**.
- *Search Space:* If the search space is narrow, hyperparameter tuning may never find the optimal configuration which resolves conflicts. It is possible that hyperparameter tuning did

not work for some combinations as the search space was not broad (e.g., **C17**, **C32**, **C35**).
- *Fundamental Incompatibility:* Some defenses are fundamentally incompatible and cannot be resolved by hyperparameter tuning. I discuss this further below under "Limitations".
- *Optimization Landscape:* If the loss landscape contains many local minima, hyperparameter tuning may not find the optimal configuration to resolve conflicts. A practitioner can try more sophisticated hyperparameter tuning such as randomized search or Bayesian optimization, to see if it resolves conflicts.

Limitations: When hyperparameter tuning fails to resolve a conflict, it is not possible to conclusively mark the combination as a conflict: despite considering a feasible search space in the evaluation, some conflict-resolving hyperparameters may have been missed given the vast number of possibilities. Also, when a combination is marked as conflict (e.g., "**S4**=Yes" for **C21**, **C36**, **C37**, and **C38**), it suggests an *empirical incompatibility* among defenses but does not imply a conclusive proof or *fundamental incompatibility*. Establishing fundamental incompatibility requires a theoretical analysis which is left as future work (Chapter 9: Section 9.3.1).

4.5.5 Other Requirements

I now discuss how Def\Con meets scalability (**R2**), non-invasive (**R3**), and generality (**R4**).

Scalability (R2): None of the prior works have considered more than two defenses. Since Def\Con allows for applying defenses in three stages of the ML pipeline, it should theoretically support at least three defenses. To illustrate this, I follow the instructions in Section 4.6 to extend Def\Con beyond two defenses. I begin with pairwise combinations predicted as effective (marked as Δ in Tables 4.5 and 4.6), which align with empirical evaluation, and then include additional defenses to check for pairwise conflicts. I consider five combinations with three defenses each, which should be effectively combined (marked as Δ). I report the results in Table 4.8 and find that it is indeed possible to effectively combine three defenses using Def\Con (**R2**). These are illustrative examples to show that Def\Con is scalable to more than two defenses. The goal here is not an exhaustive evaluation of the large number of multi-way combinations, but only to show that effective multi-way combinations which are previously unexplored, are possible.

Non-Invasive (R3): Def\Con extends **T2** and hence, inherits the non-invasive requirement. I use existing defenses proposed in the literature without modifying them, and only adapting them to the datasets. *In summary, Def\Con satisfies **R3**.*

General (R4): Def\Con identifies a conflict based on (a) the relative position of the defenses in the ML pipeline, and (b) the mechanisms that underlie them (whether one defense uses a risk

Table 4.8: **Scalability (R2) of Def\Con to $\texttt{Def}_1$, $\texttt{Def}_2$, and $\texttt{Def}_3$ (in order):** Color coding and notations are the same as in Tables 4.5 and 4.6.

	Combinations	Metric	FMNIST	UTKFACE
C39	$\texttt{Def}_1$: EvsnRob-In $\texttt{Def}_2$: Expl-Post $\texttt{Def}_3$: MdlWM-Post	acc (↑) robacc (↑) err (↓) wmacc (↑)	87.38 ± 0.15 79.37 ± 0.29 0.96 ± 0.14 100.00 ± 0.00	74.34 ± 0.72 39.21 ± 0.32 0.17 ± 0.05 73.33 ± 8.83
C40	$\texttt{Def}_1$: PoisnRob-In $\texttt{Def}_2$: Expl-Post $\texttt{Def}_3$: MdlWM-Post	acc (↑) ASR (↓) err (↓) wmacc (↑)	89.47 ± 0.24 9.81 ± 0.12 0.06 ± 0.02 100.00 ± 0.00	79.42 ± 0.51 66.74 ± 12.11 0.52 ± 0.04 77.14 ± 11.82
C41	$\texttt{Def}_1$: PoisnRob-Post $\texttt{Def}_2$: Expl-Post $\texttt{Def}_3$: MdlWM-Post	acc (↑) ASR (↓) err (↓) wmacc (↑)	89.47 ± 0.24 9.81 ± 0.12 0.06 ± 0.02 100.00 ± 0.00	67.04 ± 3.35 1.85 ± 3.39 0.17 ± 0.10 81.90 ± 7.00
C42	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: GpFair-In $\texttt{Def}_3$: Expl-Post	acc (↑) wmacc (↑) eqodds (↓) err (↓)		77.53 ± 1.75 100.00 ± 0.00 0.00 ± 0.00 0.01 ± 0.00
C43	$\texttt{Def}_1$: DtWM-Pre $\texttt{Def}_2$: GpFair-In $\texttt{Def}_3$: MdlWM-Post	acc (↑) RSD (↑) eqodds (↓) wmacc (↑)		79.17 ± 0.93 100.00 ± 0.00 0.00 ± 0.00 73.33 ± 7.12
C44	$\texttt{Def}_1$: GpFair-In $\texttt{Def}_2$: PoisnRob-Post $\texttt{Def}_3$: Expl-Post	acc (↑) eqodds (↓) ASR (↓) err (↓)		69.42 ± 2.09 8.12 ± 4.49 0.13 ± 0.25 0.05 ± 0.02

that is being protected by a later defense). Def\Con does not rely on specific defenses. Hence, Def\Con is likely to be applicable beyond the initial set of defenses I used for evaluation (i.e., commonly available defenses from the literature). For identifying conflicts of a new defense with others, a practitioner can first identify its position in the ML pipeline, and determine whether it uses a risk that is defended by a later defense. Since these are independent of any specific defense, Def\Con is applicable for any defense that can be mapped to Def\Con's flowchart (Figure 4.1). I select specific defense implementations based on their availability (see Section 4.4.4). Other implementations can be used without affecting the conclusions.

Takeaway: Def\Con scales beyond two defenses (R2), is non-invasive (R3) and general (R4).

4.6 Discussions

Note on Model Utility: So far, I have focused only on the *defense effectiveness* by examining how combining defenses impacts the effectiveness of each individual defense. An additional pre-requisite for deploying a defense combination is whether it negatively impacts *model utility*. I define a defense combination to be *viable* if it is (a) effective and (b) incurs only a minimal utility drop compared to the lowest of the "single defense" baseline. In Tables 4.5 and 4.6, I observe that all the combinations which `Def\Con` predicted as effective are also viable. For **C15**, **C27**, and **C30**, the utility is worse than the "single defense" baseline. These were already flagged as ineffective. I did not observe any combinations that are effective but not viable.

Extending `Def\Con` to predict the viability of the combinations is challenging, since it is unclear how a defense impacts model utility. There are two possible cases:

- For both defenses, if utility is either better or similar to the "single defense" baseline, it is likely that the combination will have acceptable utility.
- If the utility degrades for both defenses, the combination is likely to have poor utility and hence non-viable. However, it is also possible (as seen in Tables 4.5 and 4.6) that the utility of the combination does not fall below the "minimum utility for single defenses" baseline, even if some, but not all, constituent defenses fall below their respective "no-defense" baseline. It is unclear what mechanisms account for this phenomenon.

Hence, quantifying the impact of individual defenses on utility is an open problem for several defenses (e.g., adversarial training [135, 138, 398, 399, 294] and differential privacy [183, 400, 401, 402]), and an area of active research.

Extending Beyond Two Defenses (Multi-way Combinations): To extend `Def\Con` beyond two defenses, the algorithm decomposes the multi-way combination into pairwise comparisons, invoking the original `Def\Con` flowchart (Figure 4.1) to check for conflicts. Determining conflict for multi-way combinations from pairwise combinations is principled and not limiting. There are two cases when decomposing multi-way combinations into pairwise combinations:

- **Case 1: Defenses in different stages:** When combining three defenses across stages, conflicts can be checked pairwise since the application order is fixed; any detected pairwise conflict is treated as an overall conflict (marked as Δ).
- **Case 2: More than one defense in one stage:** All possible permutations of the defenses in a given stage must be checked, and determine whether there is a conflict. If there is no conflict, we combine with other stage defenses, and check for the conflicts.

I now present the algorithm to evaluate conflict for multi-way combinations, and assume that we have a set of defenses—partitioned into three ordered stages. The algorithm iterates through each stage in order, and skips stages with no defenses:

- **If the stage has single defense:** For a stage s, I consider the defense as $\texttt{Def}_s$ ($s \in [1, 3]$) and check its compatibility with defenses in other stages.
- **If the stage has multiple defenses:** For each such stage:
  - **Permutation Checking:** Consider all permutations of the defenses. For each permutation, check every consecutive pair of defenses using Def\Con flowchart to detect conflicts.
  - **Pruning and Selection**: If a pair causes a conflict, discard that permutation and prune others containing that conflicting pair (e.g., for defenses: A, B, and C, if "AB" conflict, then no need to check for "ABC" or "CAB"). If a conflict-free permutation is found, treat the entire sequence as a single composite defense, $\texttt{Def}_s$ (e.g., say "CAB" does not conflict).
- **Sequential Composition**: Treat the resulting composite defense as an atomic unit (e.g., $\texttt{Def}_1$ = "CAB") and check for conflicts with the next stage's resolved defenses (e.g., $\texttt{Def}_2$ = "DE"), using Def\Con flowchart between $\texttt{Def}_1$ and $\texttt{Def}_2$. I check whether any of the constituent defenses in $\texttt{Def}_1$ use a risk mitigated by defenses in $\texttt{Def}_2$ (**S3** in Figure 4.1).
- **Termination:** If any invocation of Def\Con for pairwise check indicates a conflict, terminate and report as a conflict (marked as Δ), else, as alignment (marked as Δ).

Other Combination Techniques: In Chapter 3, Tables 3.3 and 3.4 systematized unintended interactions among defenses and risks. An alternative naïve approach would be to reject combinations in which one defense increases the risks mitigated by another. However, this is overly restrictive and discards several non-conflicting combinations (e.g., Expl-Post and MdlWM, EvsnRob-In and Fngrprnt-Post). Since many interactions in Chapter 3, Tables 3.3 and 3.4, remain unexplored, it is challenging to apply this naïve approach.

Other Dataset Modalities: I extend the evaluation to tabular dataset, and not all defenses transfer to other data modalities. For example, when adapting the defenses to the CENSUS dataset, only 4 of the 11 implemented defenses were applicable: image-specific techniques like poisoning and watermarking do not apply. This gave two valid combinations: (i) evasion robustness + explanations and (ii) group fairness + explanations (others were excluded due to incompatibility). Both combinations aligned with Def\Con 's predictions.

Contribution in Support of the Thesis Statement

Existing work lacks a principled technique to combine ML defenses that is accurate, scalable, non-invasive, and general. By analyzing the limitations of prior work, I designed Def\Con which meets all requirements and predicts if (unexplored) defense combinations are effective. Def\Con accounts for the factors underlying conflicting interactions among defenses, that include where each defense is applied in the ML pipeline (pre-, in-, or post-training), and whether the underlying mechanisms interfere (e.g., a later defense minimizes a risk used by an earlier defense). Thus, I answer **RQ1.3** to support the thesis statement.

Chapter 5

Colluding Adversaries in ML Pipelines

Overview

Adversaries in an ML pipeline with different characteristics (i.e., objectives, knowledge, and capabilities) can *collude* by executing an attack to amplify others. Existing work lacks a systematic framework to explore collusion among adversaries and to study the implications of adversary's characteristics. I present a framework covering collusion (a) between train-time and inference-time adversaries, and (b) among inference-time adversaries. The framework includes *factors that aid collusion potential* among adversaries. I propose a guideline to conjecture about the potential for collusion[a] using these enabling factors. I use the guideline to explain prior work, conjecture about unexplored collusion, and empirically validate five such cases. Finally, I discuss how adversaries' characteristics influence the potential for collusion. Overall, this work identifies factors enabling collusion among adversaries, thus, answering **RQ1.3** to support the thesis statement.

[a]**Terminology:** Here, I study the "potential for collusion" among adversaries but for brevity, I refer to it as "collusion", and use them interchangeably.

So far, in Chapters 3 and 4, I have considered adversaries ($\mathcal{A}dv$) with different characteristics (i.e., objectives, knowledge, and capabilities) who exploit different risks[1] as independent

[1]**Terminology**: $\mathcal{A}dv$ executes an attack by exploiting a vulnerability corresponding to a risk (e.g., Shokri et al. [266] proposed a specific attack for `MemInf` which exploits the vulnerability of distinguishability of model observables across different datasets). In Chapter 3, I examined how defenses influence susceptibility to risks more broadly, as changes in susceptibility impact the effectiveness of various attacks that exploit those risks. Here, I focus on how executing an attack can increase the effectiveness of others. For simplicity, I use the notations for risks from Chapter 2, Section 2.3 (e.g., `Evsn`, `MemInf`) to refer to attacks.

of each other. However, multiple adversaries in the ML pipeline either with different roles (e.g., model trainer and client) or with the same role (e.g., clients with different objectives), can *collude* by executing one attack to increase the effectiveness of others. Collusion is defined as the explicit coordination among different adversaries in the ML pipeline to execute one attack and use the outcomes to improve another. This is targeted attack composition. Collusion among adversaries with different objectives is well documented in other domains: cryptocurrency pump-and-dump schemes where one adversary inflates a stock, which can be exploited by others for profit [403]; and cyber-crime markets where an adversary sells compromised data for downstream attacks like identity theft [404]. Similar collusion within ML pipelines is feasible, and merits investigation.

For instance, consider two different adversaries: $\mathcal{A}dv1$ seeking to find an adversarial example against a target model $\mathcal{M}$ and $\mathcal{A}dv2$ seeking to build a shadow model of $\mathcal{M}$ either for some privacy attack [405], or to steal $\mathcal{M}$'s functionality [106]. $\mathcal{A}dv1$ sends multiple queries to $\mathcal{M}$ until it successfully finds an adversarial example [406], usually discarding all other query-response pairs. But, they are valuable for $\mathcal{A}dv2$. Therefore, $\mathcal{A}dv1$ can either sell the query-response data to $\mathcal{A}dv2$ or share the cost of querying $\mathcal{M}$. Hence, adversaries are incentivized to collude (a) via a marketplace for selling attack information, or (b) by sharing costs to avoid duplication of efforts.

Foreseeing all the possible cases of collusion among adversaries is challenging. A unified framework to study collusion can help *practitioners* and *researchers* to identify new threats, design strong attacks, and design defenses against such attacks. Ideally, the framework should be (i) *comprehensive* (captures $\mathcal{A}dv$'s characteristics *across multiple attacks*) and (ii) *extensible* (allows inclusion of additional attacks and $\mathcal{A}dv$'s characteristics). This allows a systematic exploration of collusion among adversaries, and study the implications of $\mathcal{A}dv$'s characteristics on collusion. Through this framework, I aim to answer: (**RQ1.3**) *what are the factors that enable collusion among adversaries, and how can we predict the potential for collusion?*

I present the *first* such framework that includes various factors enabling collusion. The framework covers two collusion types identified from prior work: (i) between train- and inference-time adversaries ($\rightarrow$ "*train-inference collusion*"), and (ii) among inference-time adversaries ($\rightarrow$ "*inference-time collusion*"). For both collusion types, I observe that a successful attack results in changing some factors as an outcome, which can influence the effectiveness of other attacks relying on those factors. However, these factors differ between both collusion types. For train–inference collusion, `Poisn` changes the *factors underlying overfitting and memorization* which can impact the effectiveness of inference-time attacks that rely on those factors. Here, the framework includes the factors underlying overfitting and memorization, which are manipulated by adversary $\mathcal{A}dv$. For inference-time collusion, *knowledge inferred* from the first

attack is treated as an outcome that satisfies the prerequisite for the second attack.[2]

I propose a guideline to conjecture about the potential for collusion using the common enabling factors from the framework. I empirically validate the guideline's conjectures for five unexplored collusion. Finally, I discuss how $\mathcal{A}dv$'s characteristics impact collusion, and its implications for the guideline. I claim the following contributions:

1. the *first framework* to systematically explore colluding adversaries using various *factors*, and a *guideline* to conjecture about the collusion; (Section 5.1)
2. a guideline to *explain collusion* in prior work identified from a *literature survey*; (Section 5.2)
3. identify *unexplored collusion* for future work, and use the guideline to conjecture about them, and empirically validate five such conjectured cases; (Section 5.3)
4. discuss the *influence* of $\mathcal{A}dv$'s characteristics on the collusion potential. (Section 5.4)

5.1 Framework to Explore Colluding Adversaries

The goal is to develop a framework to explore *collusion among adversaries*, and study the *influence of $\mathcal{A}dv$'s characteristics* on collusion potential. I identify the following requirements for an ideal framework: **R1 Comprehensive** (captures $\mathcal{A}dv$'s characteristics *across multiple attacks, model types, and settings*); and **R2 Extensible** (supports new attacks and characteristics).

I present the *first such framework* which includes: (i) factors enabling collusion among adversaries (Section 5.1.1), (ii) relation between attacks and factors (Section 5.1.2), and (iii) a guideline to conjecture about the potential for collusion (Section 5.1.3).

5.1.1 Factors Enabling Collusion

I start with collusion among two adversaries ($\mathcal{A}dv1$ and $\mathcal{A}dv2$), and later discuss extending beyond two adversaries in Section 5.5. I assume that $\mathcal{A}dv1$ and $\mathcal{A}dv2$ execute attacks $\texttt{Att}_1$ and $\texttt{Att}_2$, respectively. Based on prior work (see Section 5.2), I identify two collusion types:

- **Train-inference Collusion:** $\mathcal{A}dv1$ executes $\texttt{Att}_1$ (=`Poisn`) to increase the effectiveness of $\texttt{Att}_2$ for $\mathcal{A}dv2$ during inference-time.
- **Inference-time Collusion:** $\mathcal{A}dv1$ executes $\texttt{Att}_1$ during inference-time, and uses the knowledge inferred to amplify inference-time $\texttt{Att}_2$ for $\mathcal{A}dv2$.

I discuss the factors enabling these two collusion types below.

[2]**Terminology:** The factors for train-inference collusion relate to overfitting and memorization (as in Chapter 3), whereas those for inference-time collusion concern $\mathcal{A}dv$'s knowledge. For simplicity, I refer to both as factors.

Train-Inference Collusion

I conjecture that train-time attacks, i.e., Poisn, can be optimized to influence various factors underlying overfitting and memorization as an outcome (e.g., increase tail length of $\mathcal{D}_{tr}$'s distribution, distinguishability in model observables across subgroups or datasets). This can in turn affect the effectiveness of inference-time attacks that rely on those factors. Concretely, $\mathcal{A}dv1$ can optimize Poisn to increase memorization and thereby the distinguishability between (a) training and non-training data records (to increase effectiveness of MemInf [232]), (b) across demographic groups (to increase effectiveness of AttInf [232]), or (c) across datasets with different distributions (to increase effectiveness of DistInf [274]).

The framework includes factors underlying overfitting and memorization that can be manipulated by $\mathcal{A}dv1$ through Poisn, and exploited during inference-time by $\mathcal{A}dv2$. I start with the factors identified in Chapter 3, and select those that $\mathcal{A}dv1$ can control by optimizing Poisn or by manipulating the training process (e.g., objective function). I exclude the remaining factors such as (i) those related to hyperparameters which cannot be controlled by $\mathcal{A}dv1$ for a fixed experimental setup (e.g., $\mathcal{D}_{tr}$ size: D1, number of attributes: D3, model capacity: M1)[3]; and (ii) factors which are insufficiently explored (e.g., priority of learning stable attributes: D4, distance to decision boundary: O3). Finally, the framework includes the following factors:

- D2 (**Tail length of $\mathcal{D}_{tr}$'s distribution**): Increasing the tail length of $\mathcal{D}_{tr}$'s distribution can increase the memorization of specific data records by $\mathcal{M}$, thereby increasing the effectiveness of attacks that exploit long-tailed distribution. $\mathcal{A}dv1$ controls this factor by injecting Poisn (typically outliers or rare data records) to extend the tail of $\mathcal{D}_{tr}$'s distribution and thus, influence the effectiveness of other attacks (Chapter 3: Section 3.1).
- O2.1 (**Distinguishability in model observables across datasets**): $\mathcal{M}$ behaves differently on data records inside and outside $\mathcal{D}_{tr}$ [232], or when trained on datasets with different distributions [274, 275]. This distinguishability can increase the effectiveness of other attacks (e.g., MemInf and DistInf). $\mathcal{A}dv1$ can control this distinguishability by optimizing Poisn.
- O2.2 (**Distinguishability in model observables across demographic groups**): $\mathcal{M}$ behaves differently on different demographic groups (e.g., race or sex). This distinguishability can increase the effectiveness of other attacks (e.g., AttInf, DistInf, Discr). $\mathcal{A}dv1$ can control this distinguishability by optimizing Poisn [309].

[3]For poisoning attacks, neither DtOwnr nor DtProv can access or modify model- or data-related hyperparameters without detection. Although ModProv may introduce architectural backdoors, it must preserve model capacity to avoid detection. Thus, $\mathcal{A}dv1$ cannot meaningfully manipulate hyperparameters for Poisn. For inference-time attacks, Clnt, acting as $\mathcal{A}dv1$, does not control hyperparameters. I omit hyperparameters as factors.

Inference-time Collusion

$\mathcal{A}dv$ can increase their knowledge about $\mathcal{M}$ and $\mathcal{D}_{tr}$ by executing an attack to obtain the intended outcomes (e.g., distribution of $\mathcal{D}_{tr}$ from `DistInf`, or functionally similar "shadow model" $\mathcal{M}^{shdw}$[4] using `ModExt`), or some *side information* (e.g., adversarial examples as $\mathcal{D}^{aux}$ from `Evsn`). This knowledge acts as a prerequisite for later attacks (e.g., knowing $\mathcal{D}_{tr}$'s distribution or $\mathcal{M}$'s functionality can improve the effectiveness of other attacks such as `MemInf`, `AttInf`, `DistInf`). There is a potential for collusion if there exist a pair of attacks where the outcomes from $\texttt{Att}_1$ can meet the prerequisites of $\texttt{Att}_2$, that were not met otherwise.

`Te1-Aux` (**Quality of auxiliary dataset** $\mathcal{D}^{aux}$): Attacks that require training $\mathcal{M}^{shdw}$ to mimic $\mathcal{M}$, or an attack model for a specific objective, benefit from having a high-quality $\mathcal{D}^{aux}$ that overlaps with $\mathcal{D}_{tr}$. In practice, $\mathcal{D}^{aux}$ typically does not overlap with $\mathcal{D}_{tr}$. If the overlap with $\mathcal{D}_{tr}$ or the quality of $\mathcal{D}^{aux}$ increases, the attack effectiveness also increases (e.g., [125]).
`Te2-Dist` (**Similarity of** $\mathcal{D}^{aux}$**'s distribution with** $\mathcal{D}_{tr}$): Attacks that require training $\mathcal{M}^{shdw}$ benefit from having $\mathcal{D}^{aux}$'s distribution to be similar to $\mathcal{D}_{tr}$. If the distribution similarity between $\mathcal{D}^{aux}$ and $\mathcal{D}_{tr}$ increases, the attack is more effective (e.g., [407]).
`Te3-Shdw` (**Functional similarity between** $\mathcal{M}^{shdw}$ **and** $\mathcal{M}$): Attacks that require training $\mathcal{M}^{shdw}$ should ensure that its functionality mimics that of $\mathcal{M}$ to achieve better attack transferability. Hence, if $\mathcal{A}dv$'s knowledge about $\mathcal{M}$'s functionality increases, $\mathcal{M}^{shdw}$ will be more similar to $\mathcal{M}$, thereby increasing the attack effectiveness (e.g., [407]).
`Te4-Arch` (**Architecture similarity between** $\mathcal{M}^{shdw}$ **and** $\mathcal{M}$): In real-world settings, $\mathcal{A}dv$ may not know $\mathcal{M}$'s architecture (e.g., ChatGPT). If $\mathcal{A}dv$'s knowledge of $\mathcal{M}$'s architecture increases, $\mathcal{M}^{shdw}$ can be made more similar to $\mathcal{M}$, thereby increasing functional similarity and improving attack effectiveness and transferability (e.g., [407]).

The factors considered here focus on $\mathcal{A}dv$'s knowledge—specifically, what knowledge is gained from a successful attack ("outcomes") and what information is required to carry out an attack ("prerequisites"). Better knowledge of these four factors enables $\mathcal{A}dv$ to more accurately approximate $\mathcal{M}$'s behavior and design more effective attacks that transfer to $\mathcal{M}$.

These factors differ from those for train–inference collusion (and those in Chapter 3), which do not account for $\mathcal{A}dv$'s knowledge. Instead, those factors emphasize underlying effects on overfitting and memorization due to `Poisn` (or defenses in Chapter 3) and other risks. Table 5.1 summarizes factors enabling both the train-inference and inference-time collusion.

[4]For some attacks (e.g., `Evsn`, `MemInf`, `AttInf`, `DistInf`, `DtRecon`), $\mathcal{A}dv$ typically trains shadow models to mimic $\mathcal{M}$, enabling local optimization of the attack. These shadow models can be trained on $\mathcal{A}dv$'s auxiliary data $\mathcal{D}^{aux}$ either independently or derived via `ModExt`, in which case $\mathcal{M}^{shdw}$ is a surrogate model ($\mathcal{M}^{sur}$).

Table 5.1: **Factors enabling collusion and their descriptions**

Factors	Description
D2 (Chp. 3)	Tail length of $\mathcal{D}_{tr}$'s distribution
O2.1(Chp. 3)	Distinguishability of observables across datasets
O2.2 (Chp. 3)	Distinguishability of observables across demographic groups
Te1-Aux	Quality of Auxiliary Dataset $\mathcal{D}^{aux}$
Te2-Dist	Similarity of $\mathcal{D}^{aux}$'s Distribution with $\mathcal{D}_{tr}$
Te3-Shdw	Functional Similarity between $\mathcal{M}^{sur}$ and $\mathcal{M}$
Te4-Arch	Architecture Similarity between $\mathcal{M}^{sur}$ and $\mathcal{M}$

5.1.2 Relation between Attacks and Factors

I will now describe the relation between each attack and the various factors for train-inference and inference-time collusion, in terms of (i) how attack effectiveness leads to change in factors; and (ii) how changes in factors correlate with attack effectiveness.

- **Train-inference Collusion:** Table 5.2 shows (i) how the effectiveness of the train-time attack (i.e., Poisn) leads to change in factors (left half), and (ii) how changes in factors correlate with the effectiveness of various attacks (right half). For Poisn (in Table 5.2: left half), I indicate how its effectiveness correlates with changes in the factors, rather than how changes in the factors influence attack effectiveness (as done in Chapter 3). Hence, these correlations differ from those studied in Chapter 3. On the other hand, the correlation between changes in the factors and the effectiveness of various attacks remains the same as in Chapter 3. I reproduce them here for convenience (Table 5.2: right half).
- **Inference-time Collusion:** Table 5.3 shows (i) how the effectiveness of inference-time attacks leads to changes in factors (left half), and (ii) how changes in factors correlate with the effectiveness of other inference-time attacks (right half).

In both collusion types, I use $\uparrow$ (or $\downarrow$) to indicate positive (negative) correlation (Tables 5.2 and 5.3). Refer to Table 5.2 for train-inference collusion, and Table 5.3 for inference-time collusion. For each attack, I describe the relation between the attacks and the enabling factors for train-inference and inference-time collusion below.

Poisoning (Poisn)

This is a train-time attack, and hence, I only discuss the factors enabling train-inference collusion. Below, I indicate how the effectiveness of Poisn correlates with a change in the factors

Table 5.2: **Relation of attacks and factors for train-inference collusion:** I use ↑ (or ↓) to indicate positive (negative) correlation. Rows for `Poisn` under "How effectiveness of attack (`Poisn`) leads to change in factors" (left half) indicates how `Poisn` influences a change in factors, instead of how change in factors influence effectiveness of `Poisn` (as in Chapter 3). Rows under "How changes in factors correlate with attack effectiveness" (right half) are reproduced from Chapter 3; only the factors that can be manipulated by $\mathcal{A}dv$ are considered. Gray → not applicable; empty cells → no prior literature reported the correlations.

Factors → / Attack ↓	How effectiveness of attack (`Poisn`) leads to change in factors			How changes in factors correlate with attack effectiveness		
	D2 (Tail length)	D2.1 (Datasets)	D2.2 (Subgroups)	D2 (Tail length)	D2.1 (Datasets)	D2.2 (Subgroups)
`Poisn`	↑ [408]	↑ [232]	↑ [309]			
`Evsn`				↑ [409, 410]		
`MemInf`				↑ [267, 76]	↑ [411, 412]	
`AttInf`				↑ [115]		↑ [256]
`DistInf`				↑ [275, 274]	↑ [12]	↑ (Chp. 3)
`DtRecon`				↑ [413]	↑ [414]	↑ (Chp. 3)
`ModExt`				↓ [415]		
`Discr`				↑ [124]		↑ [256]

(in Table 5.2: left half under "How attack effectiveness leads to change in factors").

- ***Train-Inference Collusion:*** $\mathcal{A}dv$ increases the tail length of $\mathcal{D}_{tr}$'s distribution by adding poisons (D2: ↑) [408]. The attack can be optimized to increase the distinguishability of model observables between (a) data records inside and outside $\mathcal{D}_{tr}$ (D2.1 : ↑) [232], (b) different distributional properties in $\mathcal{D}_{tr}$ (D2.1 : ↑) [274, 275], and (c) demographic groups (D2.2: ↑) [309].

For the remaining attacks, in the case of train–inference collusion, I indicate how changes in the factors (due to `Poisn`) correlate with changes in attack effectiveness (Table 5.2: right half). In the case of inference-time collusion, I indicate both: (i) the correlation between attack effectiveness and changes in the factors (Table 5.3: left half); and (ii) the correlation between changes in the factors and attack effectiveness (Table 5.3: right half).

Table 5.3: **Relation of attacks and factors for inference-time collusion:** I use ↑ (or ↓) to indicate positive (negative) correlation. I exclude `Discr`, as it does not correspond to any attack that $\mathcal{A}dv$ can exploit. Note that the factors here are related to the $\mathcal{A}dv$'s *knowledge* inferred from the first attack (as outcomes) that satisfies the prerequisite for the second.

	How attack effectiveness leads to change in factors (as "outcomes")				How change in factors (as "prerequisites") correlate with attack effectiveness			
Factors→ Attack↓	**Te1-Aux (Aux. Data)**	**Te2-Dist (Data Dstrb.)**	**Te3-Shdw (Model Fnct.)**	**Te4-Arch (Model Arch.)**	**Te1-Aux (Aux. Data)**	**Te2-Dist (Data Dstrb.)**	**Te3-Shdw (Model Fnct.)**	**Te4-Arch (Model Arch.)**
`Evsn`	↑ [108, 411]				↑ [416, 417, 418]	↑ [417, 416, 418]	↑ [417, 416, 418]	↑ [417, 416, 418]
`MemInf`	↑ [266, 407]				↑ [266, 407]	↑ [266, 407]	↑ [266, 407]	↑ [266, 407]
`AttInf`		↑ [256]			↑ [115, 278]	↑ [115, 278]	↑ [278]	↑ [278]
`DistInf`		↑ [12, 121]			↑ [12, 121]		↑ [12, 121]	↑ [12, 121]
`DtRecon`	↑ [49, 118]				↑ [49, 118]	↑ [49, 118]	↑ [49, 118]	↑ [49, 118]
`ModExt`			↑ [106, 108]	↑ [106, 108]	↑ [106, 108]	↑ [106, 108]		

Evasion (`Evsn`)

• ***Train-Inference Collusion:*** As the tail length of $\mathcal{D}_{tr}$ increases, models are more susceptible to adversarial examples particularly from the tail classes (D2: ↑) [226, 410]. This is because models perform well on well-represented classes (from the head of the distribution), but struggle to classify under-represented classes (from the tail of the distribution).

• ***Inference-time Collusion:*** In a blackbox setting, $\mathcal{A}dv$ has to train $\mathcal{M}^{shdw}$ to find transferable adversarial examples. The effectiveness of such adversarial examples is positively correlated with the quality of $\mathcal{D}^{aux}$ used to train $\mathcal{M}^{shdw}$ (`Te1-Aux`: ↑), similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (`Te2-Dist`: ↑), and similarity of $\mathcal{M}^{shdw}$'s functionality and architecture with $\mathcal{M}$ (`Te3-Shdw`: ↑, `Te4-Arch`: ↑) [417, 416, 418]. Successful `Evsn` provides adversarial examples as *side information* that can be used as $\mathcal{D}^{aux}$ for other attacks (`Te1-Aux`: ↑) [108, 411, 419]. Higher effectiveness of `Evsn` results in improving the quality of $\mathcal{D}^{aux}$.

Membership Inference (MemInf)

• ***Train-Inference Collusion:*** As the tail length increases, outlier data records (from the tail classes) are memorized more, resulting in a higher effectiveness of MemInf (D2: ↑) [267, 76]. Furthermore, the effectiveness of MemInf increases with higher distinguishability in model observables across data records inside and outside $\mathcal{D}_{tr}$ (O2.1 : ↑) [266].
• ***Inference-time Collusion:*** The effectiveness of MemInf is positively correlated with the quality of $\mathcal{D}^{aux}$ (Te1-Aux: ↑), similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (Te2-Dist: ↑), and similarity of $\mathcal{M}^{shdw}$'s functionality and architecture with $\mathcal{M}$ (Te3-Shdw, Te4-Arch: ↑) [266, 407]. After a successful attack, $\mathcal{A}dv$ identifies data records in $\mathcal{D}_{tr}$, which can act as $\mathcal{D}^{aux}$ for other attacks (Te1-Aux: ↑) [266, 407]. The accuracy of identifying data records in $\mathcal{D}_{tr}$ increases with MemInf's effectiveness.

Attribute Inference (AttInf)

• ***Train-Inference Collusion:*** AttInf is more effective as $\mathcal{D}_{tr}$'s tail length increases (D2: ↑) [115]. AttInf exploits the distinguishability in model observables across demographic groups. Hence, increasing distinguishability makes AttInf effective (O2.2: ↑) [256].
• ***Inference-time Collusion:*** The effectiveness of AttInf is positively correlated with quality of $\mathcal{D}^{aux}$ (Te1-Aux: ↑) [115, 278], similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (Te2-Dist: ↑) [115, 278], and similarity of $\mathcal{M}^{shdw}$'s functionality and architecture with $\mathcal{M}$ (Te3-Shdw, Te4-Arch: ↑) [278]. AttInf can help estimate $\mathcal{D}_{tr}$'s distribution when evaluated across several data records sampled i.i.d from $\mathcal{D}_{tr}$ (Te2-Dist: ↑) [115]. The estimate about $\mathcal{D}_{tr}$'s distribution is better with more effective AttInf [420, 256, 115].

Distribution Inference (DistInf)

• ***Train-Inference Collusion:*** As the tail length increases, data records from the tail classes are memorized more, resulting in higher DistInf (D2: ↑) [275, 274]. DistInf exploits the distinguishability of model observables when trained on datasets with different properties, and across different subgroups. Hence, increasing this distinguishability makes DistInf more effective (O2.1, O2.2: ↑) [12] [Chapter 3].
• ***Inference-time Collusion:*** The effectiveness of DistInf is positively correlated with quality of $\mathcal{D}^{aux}$ (Te1-Aux: ↑), and similarity of $\mathcal{M}^{shdw}$'s functionality and architecture with $\mathcal{M}$ (Te3-Shdw, Te4-Arch: ↑) [12, 121]. After a successful attack, $\mathcal{A}dv$ learns $\mathcal{D}_{tr}$'s distribution (Te2-Dist: ↑). Estimating $\mathcal{D}_{tr}$'s distribution is better with more effective DistInf [12, 121].

Data Reconstruction (DtRecon)

• ***Train-Inference Collusion:*** Data records from tail classes are more susceptible to DtRecon (D2: ↑) [413]. Higher distinguishability across datasets and demographic groups results in better attack effectiveness (O2.1 , O2.2: ↑) [414].
• ***Inference-time Collusion:*** The effectiveness of DtRecon is positively correlated with the quality of $\mathcal{D}^{aux}$ (Te1-Aux: ↑), the similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (Te2-Dist: ↑), and the similarity of $\mathcal{M}^{shdw}$'s functionality and architecture with $\mathcal{M}$ (Te3-Shdw, Te4-Arch: ↑) [49, 118]. After a successful attack, $\mathcal{A}dv$ infers data records in $\mathcal{D}_{tr}$, which can act as $\mathcal{D}^{aux}$ for other attacks (Te1-Aux: ↑) [117, 49, 118, 86, 87]. $\mathcal{A}dv$ can more accurately infer data records in $\mathcal{D}_{tr}$ as the effectiveness of DtRecon increases.

Unauthorized Model Ownership (ModExt)

• ***Train-Inference Collusion:*** As the tail length increases, $\mathcal{M}$ may not sufficiently capture the tail classes. Hence, training $\mathcal{M}^{shdw}$ via model extraction of $\mathcal{M}$ exacerbates the utility drop on tail classes for $\mathcal{M}^{shdw}$ (D2: ↓) [415].
• ***Inference-time Collusion:*** Effectiveness of ModExt is positively correlated with the quality of $\mathcal{D}^{aux}$ (Te1-Aux: ↑), and the similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (Te2-Dist: ↑) [106, 108]. $\mathcal{A}dv$ can either learn $\mathcal{M}$'s architecture (Te4-Arch: ↑) [421, 422], or obtain a functionally similar $\mathcal{M}^{shdw}$ (Te3-Shdw: ↑) [106, 108]. More effective attacks can extract the architecture and functionality of $\mathcal{M}$ more accurately.

Discriminatory Behavior (Discr)

Recall that Discr is a risk which does not have an attack. Accordingly, I consider only train–inference collusion (to study collusion scenarios in which executing train-time attack Poisn increases Discr), and exclude Discr from inference-time collusion, where there is no notion of "$\mathcal{A}dv$ executing attacks to exploit Discr".

• ***Train-Inference Collusion:*** As the tail length increases (addition of new minority groups), the effectiveness of Discr also increases (D2: ↑) [124]. Since Discr measures distinguishability across demographic groups, higher distinguishability implies higher Discr (O2.2: ↑) [256].

5.1.3 Guideline to Predict Collusion

For both train–inference and inference-time collusion, I conjecture that one can predict the potential for collusion by analyzing the common factors between two attacks (i.e., comparing

the left half under "How attack effectiveness leads to change in factors" and the right half under "How changes in factors correlate with attack effectiveness" in Tables 5.2 and 5.3). However, defining an exact algorithm is challenging due to the complex and only partially understood dynamics of various factors. Therefore, I present a guideline using expert insight to predict if there is potential for collusion.

For the guideline, I consider two adversaries $\mathcal{A}dv1$ and $\mathcal{A}dv2$ who execute $\texttt{Att}_1$ and $\texttt{Att}_2$, respectively. For train-inference collusion, $\texttt{Att}_1$ is always `Poisn` while $\texttt{Att}_2$ is an inference-time attack. For inference-time collusion, both $\texttt{Att}_1$ and $\texttt{Att}_2$ are inference-time attacks.

G1 Identify the common factor(s) <$\texttt{f}_c$> by matching the non-empty entries across the two halves of the table: compare columns where the row corresponds to $\texttt{Att}_1$ on the left (under "How attack effectiveness leads to change in factors") with the same columns where the row corresponds to $\texttt{Att}_2$ on the right (under "How changes in factors correlate with attack effectiveness"). See Tables 5.2 and 5.3 for train–inference and inference-time collusion respectively.
G2 If <$\texttt{f}_c$> is available, indicate the common factors using a pair of arrows in the format "<$\texttt{f}_c$>: (<·>$_1$, <·>$_2$)" where <·>$_1$ corresponds to the arrow depicting "How attack effectiveness leads to change in factors" (for $\texttt{Att}_1$) while <·>$_2$ corresponds to the arrow depicting "How changes in factors correlate with attack effectiveness" (for $\texttt{Att}_2$).

- If arrows are aligned (↑, ↑), it suggests that $\mathcal{A}dv1$ and $\mathcal{A}dv2$ can potentially collude as exploiting $\texttt{Att}_1$ increases $\texttt{Att}_2$.
- If arrows are not aligned (e.g., ↑, ↓), it suggests $\mathcal{A}dv1$ and $\mathcal{A}dv2$ are unlikely to collude (e.g., exploiting $\texttt{Att}_1$ either decreases $\texttt{Att}_2$ or does not meet prerequisites of $\texttt{Att}_2$).

G3 If <$\texttt{f}_c$> is not available, one cannot conclude about collusion potential.

I use the guideline to explain collusion identified in prior work (Section 5.2), conjecture about unexplored ones, and empirically validate one such case (Section 5.3). I later discuss how to account for $\mathcal{A}dv$'s characteristics (Section 5.4), and extend beyond two adversaries (Section 5.5).

5.2 Systematization of Prior Work

I apply the guideline to explain collusion in prior work using the factors from the framework (Section 5.2.1 and Section 5.2.2). I identify prior work on collusion through a literature survey.

Methodology for Collecting Papers: I surveyed papers covering colluding adversaries starting with an initial set of papers using keywords containing both attacks. For instance, "poisoning or backdoors increase <$\texttt{Att}_2$>" for train-inference collusion, and "$\texttt{Att}_1$ increases $\texttt{Att}_2$" for inference-time collusion. I then reviewed the citations of these papers for comprehensive coverage. Overall, I collected 28 papers for train-inference and 12 for inference-time collusion.

5.2.1 Train-Inference Collusion

Table 5.4 shows train-inference collusion (indicated as "$\texttt{Att}_1 \rightarrow \texttt{Att}_2$" where $\texttt{Att}_1$ is always Poisn). I indicate the common enabling factors from the guideline as (↑, ↑) corresponding to ($<\cdot>_1$, $<\cdot>_2$) where $<\cdot>_1$ corresponds to the arrow depicting "How attack effectiveness leads to change in factors" (for $\texttt{Att}_1$) while $<\cdot>_2$ corresponds to the arrow depicting "How changes in factors correlate with attack effectiveness" (for $\texttt{Att}_2$).

Table 5.4: **Train-inference collusions:** Collusion between $\mathcal{A}dv1$ (where $\texttt{Att}_1$ is Poisn) and $\mathcal{A}dv2$ (corresponding to $\texttt{Att}_2$). **"Factors"** → common enabling factors (from Table 5.2) after applying the guideline to check for potential for collusion (↑, ↑). Gray → no prior work.

$\texttt{Att}_2$	**References**	**Factors** ($<\texttt{Att}_1, \texttt{Att}_2>$)	**Collusion?**
Evsn	[423, 424, 425]	D2 (↑, ↑)	**Yes**
MemInf	[232, 426, 427, 292, 428]	D2, O2.1 (↑, ↑)	**Yes**
AttInf	[232, 429, 430]	D2, O2.2 (↑, ↑)	**Yes**
DistInf	[431, 275, 274, 432, 433]	D2, O2.1, O2.2 (↑, ↑)	**Yes**
DtRecon	[232, 434, 435, 436, 437, 279, 438]	D2, O2.1, O2.2 (↑, ↑)	**Yes**
ModExt	**Our Work (§5.3.1)**	D2 (↑, ↓)	**No**
Discr	[439, 311, 309, 440, 441, 442, 443]	D2, O2.2 (↑, ↑)	**Yes**

- **Poisn → Evsn**: Poisn increases $\mathcal{D}_{tr}$'s tail length which in turn increases the effectiveness of Evsn as adversarial examples from the tail classes are more effective (D2: ↑, ↑) [423, 424, 425].
- **Poisn → MemInf**: Poisn increases $\mathcal{D}_{tr}$'s tail length, and the distinguishability between data records inside and outside $\mathcal{D}_{tr}$. This increases the effectiveness of MemInf (D2, O2.1 : ↑, ↑) [232, 426, 427, 292, 428].
- **Poisn → AttInf**: Poisn increases $\mathcal{D}_{tr}$'s tail length, and the distinguishability across demographic groups. This increases the effectiveness of AttInf (D2, O2.2: ↑, ↑) [232, 429, 430].
- **Poisn → DistInf**: Poisn increases $\mathcal{D}_{tr}$'s tail length, and the distinguishability in model observables across datasets and demographic groups . This increases the effectiveness of DistInf (D2, O2.1, O2.2: ↑, ↑) [431, 275, 274, 432, 433] [Chapter 3].

- **Poisn** → **DtRecon**: Poisn make DtRecon more effective by increasing $\mathcal{D}_{tr}$'s tail length, and increasing the distinguishability in model observables across datasets and demographic groups (D2, O2.1, O2.2: ↑, ↑) [232, 434, 435, 436, 437, 279, 438].
- **Poisn** → **Discr**: Poisn increases $\mathcal{D}_{tr}$'s tail length, and the distinguishability across different demographic groups, which increases Discr (D2, O2.2: ↑, ↑) [439, 311, 309, 440, 441, 442, 443].

Table 5.5: **Inference-time collusions:** Both adversaries perform inference-time attacks: $\mathbf{Att}_1$ and $\mathbf{Att}_2$. **"Factors"** → common enabling factors (from Table 5.3) identified after applying the guideline to check for collusion potential (prerequisite of $\mathbf{Att}_2$ is satisfied by $\mathbf{Att}_1$ indicated as ↑, ↑). Gray → no prior work.

$\mathbf{Att}_1$	$\mathbf{Att}_2$	Factors (<$\mathbf{Att}_1$, $\mathbf{Att}_2$ >)	Collusion?	$\mathbf{Att}_1$	$\mathbf{Att}_2$	Factors (<$\mathbf{Att}_1$, $\mathbf{Att}_2$ >)	Collusion?
Evsn	MemInf [405] AttInf DistInf [405] DtRecon ModExt [108]	Te1-Aux (↑, ↑)	**Yes**	DistInf	Evsn MemInf [405] AttInf [405] DtRecon ModExt	Te2-Dist (↑, ↑)	**Yes**
MemInf	Evsn AttInf [265, 444] DistInf DtRecon [86, 87] ModExt [445]	Te1-Aux (↑, ↑)	**Yes**	DtRecon	Evsn MemInf AttInf DistInf ModExt [446]	Te1-Aux (↑, ↑)	**Yes**
AttInf	Evsn MemInf DistInf DtRecon ModExt	Te2-Dist (↑, ↑)	**Yes**	ModExt	Evsn [108, 447] MemInf [445] AttInf [448] DistInf DtRecon	Te3-Shdw, Te4-Arch(↑, ↑)	**Yes**

5.2.2 Inference-time Collusion

Table 5.5 shows inference-time collusion (indicated as "$\mathbf{Att}_1 \rightarrow \mathbf{Att}_2$") and the common factors among $\mathbf{Att}_1$ and $\mathbf{Att}_2$ indicated as (<·>$_1$, <·>$_2$) where <·>$_1$ corresponds to the arrow depicting "How attack effectiveness leads to change in factors" (for $\mathbf{Att}_1$) while <·>$_2$ corresponds to the arrow depicting "How change in factors correlate with attack effectiveness" (for $\mathbf{Att}_2$). The guideline explains collusion in prior work by identifying the prerequisites for $\mathbf{Att}_2$ which are satisfied by the outcomes of $\mathbf{Att}_1$ which are indicated as (↑, ↑).

- **Evsn** → {**ModExt**, **MemInf**, **DistInf**}: ModExt, MemInf, and DistInf require $\mathcal{D}^{aux}$ to train $\mathcal{M}^{shdw}$ (Te1-Aux: ↑). Evsn can provide $\mathcal{D}^{aux}$ with adversarial examples to meet the prerequisites of ModExt, MemInf, and DistInf (Te1-Aux: ↑) [108, 405].

- **MemInf** $\rightarrow$ {**AttInf**, **DtRecon**, **ModExt**}: AttInf, DtRecon, and **ModExt** can be more effective when $\mathcal{D}^{aux}$ has some overlap with $\mathcal{D}_{tr}$ to increase the functional similarity of $\mathcal{M}^{shdw}$ with $\mathcal{M}$ (Te1-Aux: $\uparrow$). $\mathcal{A}dv1$ can exploit MemInf to identify data records in $\mathcal{D}_{tr}$ which can be added to $\mathcal{D}^{aux}$ to improve the functional similarity of $\mathcal{M}^{shdw}$ to $\mathcal{M}$. This enables effective (and transferrable) attacks: AttInf, DtRecon, and **ModExt** (Te1-Aux: $\uparrow$) [265, 86, 87].
- **DistInf** $\rightarrow$ {**AttInf**, **MemInf**}: AttInf and MemInf are more effective if $\mathcal{D}^{aux}$ has a similar distribution as $\mathcal{D}_{tr}$ (Te2-Dist: $\uparrow$). By exploiting DistInf, $\mathcal{A}dv1$ can infer $\mathcal{D}_{tr}$'s distribution, and $\mathcal{A}dv2$ use that knowledge to update $\mathcal{D}^{aux}$ to train $\mathcal{M}^{shdw}$ which is functionally more similar to $\mathcal{M}$. This can improve the effectiveness of AttInf and MemInf (Te2-Dist: $\uparrow$) [405].
- **DtRecon** $\rightarrow$ **ModExt**: Training $\mathcal{M}^{shdw}$ on $\mathcal{D}^{aux}$ which partially overlaps with $\mathcal{D}_{tr}$, increases ModExt's effectiveness (Te1-Aux: $\uparrow$). DtRecon can identify data records in $\mathcal{D}_{tr}$ which can be added to $\mathcal{D}^{aux}$. This increases the overlap with $\mathcal{D}_{tr}$ improving the functional similarity of $\mathcal{M}^{shdw}$ with $\mathcal{M}$, enabling a more effective ModExt (Te1-Aux: $\uparrow$) [446].
- **ModExt** $\rightarrow$ {**Evsn**, **MemInf**, **AttInf**}: Transferable adversarial examples (in a blackbox setting), MemInf, AttInf, and DtRecon require $\mathcal{M}^{shdw}$ which is functionally similar to $\mathcal{M}$ (Te3-Shdw, Te4-Arch: $\uparrow$). ModExt can help extract such $\mathcal{M}^{shdw}$ from $\mathcal{M}$ to meet the prerequisites of other attacks (Te3-Shdw, Te4-Arch: $\uparrow$) [447, 108, 445, 448].

5.3 Unexplored Collusion among Adversaries

I now apply the guideline to conjecture about the potential of unexplored collusion (marked as gray in Tables 5.4 and 5.5). I discuss these conjectures, and empirically validate five unexplored cases of collusion (one for train-inference collusion and four for inference-time collusion)[5].

5.3.1 Conjectures for Unexplored Train-Inference Collusion

There is only one unexplored train-inference collusion between $\mathcal{A}dv1$ who executes **Poisn**, and $\mathcal{A}dv2$ who executes **ModExt** (marked as gray in Table 5.4). I identify that tail length of $\mathcal{D}_{tr}$'s distribution (D2) is the common factor between both attacks. Since the arrows are opposing ($\uparrow$ for Poisn and $\downarrow$ for ModExt), the guideline suggests that there is no potential for collusion (Poisn os likely to decrease the effectiveness of ModExt). Concretely, since Poisn increases the tail length of $\mathcal{D}_{tr}$ (D2), I conjecture that $\mathcal{M}^{sur}$ may fail to learn the under-represented data records, thereby reducing effectiveness of ModExt. I now empirically validate the conjecture.

[5]Code: https://github.com/ssg-research/sok-collusion

Table 5.6: **Poisn** → **ModExt**: Surrogate model's accuracy ("Surr. Acc.") and fidelity ("Surr. Fid.") for 0% poison are the baseline. For a given budget, green → higher than baseline, orange → similar to baseline (within std. dev.), and red → lower than baseline. I report target accuracy to show that ModExt is effective for different poisoning rates.

Budget	CIFAR10 Surr. Acc.	CIFAR10 Surr. Fid.	CIFAR100 Surr. Acc.	CIFAR100 Surr. Fid.
0% Poison (Baseline)				
	Target Acc.: $87.37_{\pm 0.16}$		**Target Acc.**: $62.24_{\pm 0.47}$	
2500	$76.83_{\pm 0.71}$	$78.54_{\pm 0.54}$	$46.99_{\pm 1.59}$	$54.75_{\pm 1.16}$
6250	$83.41_{\pm 0.23}$	$87.19_{\pm 1.09}$	$53.04_{\pm 1.07}$	$62.63_{\pm 0.92}$
12500	$85.05_{\pm 0.27}$	$88.82_{\pm 0.63}$	$54.67_{\pm 0.84}$	$65.40_{\pm 0.70}$
25000	$85.91_{\pm 0.43}$	$89.58_{\pm 1.07}$	$56.09_{\pm 0.88}$	$66.55_{\pm 0.90}$
5% Poison				
	Target Acc.: $78.19_{\pm 0.60}$		**Target Acc.**: $33.69_{\pm 1.36}$	
2500	$75.30_{\pm 1.12}$	$79.84_{\pm 1.71}$	$33.25_{\pm 0.29}$	$37.24_{\pm 0.60}$
6250	$78.04_{\pm 0.42}$	$83.31_{\pm 0.85}$	$35.00_{\pm 0.13}$	$39.90_{\pm 0.59}$
12500	$78.30_{\pm 0.25}$	$83.31_{\pm 0.31}$	$35.34_{\pm 1.21}$	$40.81_{\pm 1.22}$
25000	$78.93_{\pm 0.26}$	$84.39_{\pm 0.24}$	$34.80_{\pm 0.55}$	$40.14_{\pm 1.41}$
10% Poison				
	Target Acc.: $77.05_{\pm 0.49}$		**Target Acc.**: $33.57_{\pm 1.66}$	
2500	$74.60_{\pm 0.60}$	$78.96_{\pm 0.58}$	$32.54_{\pm 1.34}$	$36.96_{\pm 0.99}$
6250	$75.86_{\pm 0.23}$	$80.59_{\pm 0.85}$	$35.56_{\pm 1.85}$	$40.49_{\pm 1.43}$
12500	$78.29_{\pm 0.51}$	$83.95_{\pm 0.69}$	$35.00_{\pm 3.05}$	$40.11_{\pm 3.20}$
25000	$78.00_{\pm 0.30}$	$83.87_{\pm 0.53}$	$34.76_{\pm 1.34}$	$40.47_{\pm 0.87}$
15% Poison				
	Target Acc.: $75.90_{\pm 0.56}$		**Target Acc.**: $33.51_{\pm 1.95}$	
2500	$75.29_{\pm 0.61}$	$79.16_{\pm 0.22}$	$33.25_{\pm 1.23}$	$38.20_{\pm 0.87}$
6250	$75.99_{\pm 1.56}$	$81.17_{\pm 1.72}$	$34.23_{\pm 1.71}$	$39.72_{\pm 1.38}$
12500	$76.33_{\pm 1.55}$	$81.63_{\pm 0.73}$	$34.89_{\pm 1.11}$	$41.32_{\pm 0.78}$
25000	$77.11_{\pm 0.86}$	$82.54_{\pm 1.43}$	$34.99_{\pm 1.63}$	$40.98_{\pm 0.96}$
20% Poison				
	Target Acc.: $74.70_{\pm 0.40}$		**Target Acc.**: $30.38_{\pm 0.39}$	
2500	$73.03_{\pm 1.32}$	$76.86_{\pm 1.78}$	$29.78_{\pm 0.57}$	$35.25_{\pm 1.25}$
6250	$75.22_{\pm 0.70}$	$80.20_{\pm 1.74}$	$31.65_{\pm 0.97}$	$37.83_{\pm 1.27}$
12500	$75.41_{\pm 0.70}$	$81.44_{\pm 0.71}$	$31.67_{\pm 1.69}$	$38.34_{\pm 1.34}$
25000	$75.65_{\pm 0.95}$	$82.27_{\pm 0.44}$	$31.69_{\pm 1.83}$	$37.51_{\pm 1.17}$

Experimental Setup: I run the experiments on two image datasets: CIFAR 10 and CIFAR 100. The datasets contain 60,000 images (50,000 for training and 10,000 for testing) where each data point is a 32×32 colored image. The images are clustered into 10 and 100 classes for CIFAR 10 and CIFAR 100 respectively, representing different objects. I use 25,000 records for $\mathcal{D}_{tr}$ and the remaining 25,000 records as $\mathcal{D}^{aux}$ for training $\mathcal{M}^{sur}$ via ModExt. Hence, the accuracy of $\mathcal{M}$ is

not the same as the state-of-the-art on these datasets. I use ResNet models as $\mathcal{M}$ and VGG16 as $\mathcal{M}^{sur}$, and verified that $\mathcal{M}$ has been successfully poisoned using *accuracy on poisoned test dataset*[6]. I report $\mathcal{M}^{sur}$'s *accuracy* and *fidelity* (the percentage of outputs that match between $\mathcal{M}$ and $\mathcal{M}^{sur}$) on a clean $\mathcal{D}_{te}$. I use backdoors with a 5×5 white patch as the trigger, and knockoff-nets for `ModExt` by training $\mathcal{M}^{sur}$ using $\mathcal{M}$'s predictions as the ground truth [106]. I consider different query budgets for training $\mathcal{M}^{sur}$ including {10%, 25%, 50%, 100%} of the 25,000 data records in $\mathcal{D}^{aux}$.

Results: Table 5.6 shows the results of extracting $\mathcal{M}$ using `ModExt` on CIFAR 10 and CIFAR 100. For a given query budget, I compare the surrogate accuracy ("Surr. Acc.") and fidelity ("Surr. Fid.") when $\mathcal{M}$ is trained with poisons {0.05%, 0.1%, 0.15%, 0.2%}, with the case where $\mathcal{M}$ is trained without poisons (0% poison). I select a low poison rate to evaluate `ModExt`'s effect, to ensure that there is no significant utility degradation from poisoning. For a specific budget, to compare the baseline accuracy and fidelity (i.e., 0% poison), I use green for higher than baseline, orange for similar (within standard deviation), and red for lower than baseline.

Across different poison rates and budgets, the surrogate accuracy and fidelity are lower than the baseline for CIFAR 100 (red). For CIFAR 10, the surrogate accuracy and fidelity are similar to the baseline (orange) for the low budget of 2500 queries where $\mathcal{M}^{sur}$ generalizes well. However, for higher budgets, the accuracy / fidelity are lower than the baseline (red).

Takeaway: Empirical evaluation suggests that `Poisn` reduces the effectiveness of `ModExt`, thereby confirming the guideline's conjecture about the absence of collusion potential.

5.3.2 Conjectures for Unexplored Inference-time Collusion

I now discuss various unexplored inference-time collusion (gray in Table 5.5), and use the guideline to identify the potential for collusion (indicated as ↑, ↑).

- **`Evsn`** $\rightarrow$ {**`AttInf`**, **`DtRecon`**}: `Te1-Aux` is the common factor which suggests that `Evsn` generates adversarial examples which can be used as $\mathcal{D}^{aux}$ to train $\mathcal{M}^{shdw}$. Since adversarial examples can characterize $\mathcal{M}$'s decision boundary, they enable $\mathcal{M}^{shdw}$ to be functionally similar to $\mathcal{M}$. The resulting $\mathcal{M}^{shdw}$ can be used to improve the effectiveness of other attacks such as `AttInf`, `DtRecon`, and `Discr`.
- **`MemInf`** $\rightarrow$ {**`Evsn`**, **`DistInf`**}: `MemInf` can help identify data records in $\mathcal{D}_{tr}$, which can then be included as part of $\mathcal{D}^{aux}$ (`Te1-Aux`). The resulting $\mathcal{M}^{shdw}$ trained on $\mathcal{D}^{aux}$ which overlaps with $\mathcal{D}_{tr}$, is likely to increase the functional similarity with $\mathcal{M}$, thereby improving the

[6] I empirically verify this, but do not report the numbers for brevity.

effectiveness of Evsn, ModExt, and DistInf. For instance, prior work on Evsn and DistInf have shown that increasing overlap between $\mathcal{D}^{aux}$ and $\mathcal{D}_{tr}$ can increase the effectiveness of the attacks [331, 12, 449].

- **AttInf** $\rightarrow$ {**Evsn**, **MemInf**, **DistInf**, **DtRecon**,**ModExt**}: AttInf can help estimate distribution of $\mathcal{D}_{tr}$ allowing $\mathcal{A}dv$ to subsequently update $\mathcal{D}^{aux}$ (Te2-Dist). This can improve the effectiveness of Evsn, ModExt, MemInf, DtRecon, and DistInf, by training $\mathcal{M}^{shdw}$ which is functionally more similar to $\mathcal{M}$.
- **DistInf** $\rightarrow$ {**Evsn**, **DtRecon**, **ModExt**}: DistInf can infer $\mathcal{D}_{tr}$'s distribution allowing $\mathcal{A}dv$ to subsequently update $\mathcal{D}^{aux}$ (Te2-Dist). The resulting $\mathcal{M}^{shdw}$ is likely to be more similar to $\mathcal{M}$, thus, likely improving the effectiveness of Evsn, ModExt, DtRecon, and DistInf.
- **DtRecon** $\rightarrow$ {**Evsn**, **MemInf**, **AttInf**, **DistInf**}: Te1-Aux is the common factor among the attacks. DtRecon can help partially identify data records in $\mathcal{D}_{tr}$ which can augment $\mathcal{D}^{aux}$ (Te1-Aux). This can result in training $\mathcal{M}^{shdw}$ to be similar to $\mathcal{M}$, thus, improving the effectiveness of Evsn, MemInf, AttInf, and DistInf.
- **ModExt** $\rightarrow$ {**DistInf**, **DtRecon**}: ModExt can help train $\mathcal{M}^{shdw}$ that is similar to $\mathcal{M}$ (Te3-Shdw, Te4-Arch), which can be used as "shadow models" for effective DistInf and DtRecon.

Takeaway: Unexplored collusion are gaps in the literature which are left as future work.

Inference-time Collusion: ModExt $\rightarrow$DistInf

I validate the guideline's conjecture on the potential for collusion between ModExt and DistInf, marked as ($\uparrow$, $\uparrow$) in Table 5.5. Here, the common factors are Te3-Shdw and Te4-Arch suggesting that ModExt increases the knowledge about $\mathcal{M}$ for better shadow models for DistInf. I validate by comparing DistInf attack accuracy when its shadow models are trained independently on $\mathcal{D}^{aux}$ versus extracted from $\mathcal{M}$ using ModExt.

Experimental Setup: I assume both $\mathcal{A}dv1$ and $\mathcal{A}dv2$ have blackbox access to $\mathcal{M}$. $\mathcal{A}dv1$ performs ModExt where they steal the functionality of $\mathcal{M}$ by sending queries to $\mathcal{M}$ and using the corresponding outputs to train the surrogate models.

For ModExt, I use KnockoffNets [106]. For DistInf, I use the blackbox attack from Suri et al. [121] where I train an attack classifier to distinguish models trained on datasets with ratios α_1 and α_2. For example, the classifier predicts if the proportion of males in the training data is α_1 = 0.1 or α_2 = 0.9. Following Suri et al. [121], I consider two examples ratios where α_1 is close to α_2 ($\alpha_1 = 0.45$ vs. $\alpha_2 = 0.55$; and $\alpha_1 = 0.475$ vs. $\alpha_2 = 0.525$).

I use two image datasets with sensitive attributes as required for DistInf. CELEBA contains 202,599 celebrity face images with 40 binary attributes, and UTKFACE contains 23,705

in-the-wild face images with age, sex, and race attributes. I split each dataset $50/50$ into a train and a test partition, then split each partition $50/50$ (stratified jointly on the label and the sensitive attribute: sex for CELEBA, race for UTKFACE) into target and $\mathcal{A}dv$ share. I report the attack accuracy of `DistInf` as the mean and standard deviation across five runs. I confirm that fidelity and accuracy of the stolen shadow models are high and omit them for brevity.

Table 5.7: **`ModExt` → `DistInf`:** *Baseline* → shadow models trained independently without querying $\mathcal{M}$ (target); *Cross-Arch* → VGG11 shadow model extracted from $\mathcal{M}$; *Same-Arch* → ResNet34 shadow model extracted from $\mathcal{M}$. Green → higher than baseline, orange → similar to baseline (within std. dev.), red → lower than baseline.

Setting	**CELEBA**		**UTKFACE**	
	$\alpha_1 = 0.45$ $\alpha_2 = 0.55$	$\alpha_1 = 0.475$ $\alpha_2 = 0.525$	$\alpha_1 = 0.45$ $\alpha_2 = 0.55$	$\alpha_1 = 0.475$ $\alpha_2 = 0.525$
Baseline.	$70.00_{\pm 14.58}$	$55.00_{\pm 18.11}$	$55.00_{\pm 3.06}$	$45.50_{\pm 10.81}$
Cross-Arch	$98.00_{\pm 3.26}$	$96.50_{\pm 4.18}$	$91.50_{\pm 4.54}$	$88.00_{\pm 7.79}$
Same-Arch	$99.00_{\pm 1.37}$	$98.50_{\pm 3.35}$	$93.00_{\pm 7.37}$	$92.50_{\pm 4.68}$

Results: I report three settings in Table 5.7: *baseline* where the shadow models for `DistInf` are trained independently without querying $\mathcal{M}$; *Cross-Arch* where the shadow models have a different architecture than $\mathcal{M}$ (i.e., VGG11) and obtained using `ModExt`; and *Same-Arch* where the shadow model has the same architecture as $\mathcal{M}$, obtained using `ModExt`.

As expected, the attack accuracy of `DistInf` is significantly higher for cross-arch and same-arch settings (where shadow models are obtained using `ModExt`) compared to the case where the shadow models are trained independently on $\mathcal{D}^{aux}$ (in green). This confirms the conjecture that `ModExt` improves `DistInf`.

Inference-time Collusion: `DtRecon` → {`MemInf`, `AttInf`, `DistInf`}

I validate the conjecture on collusion between `DtRecon` and three privacy attacks: `MemInf`, `AttInf`, and `DistInf`, marked as (↑, ↑) in Table 5.5. The common factor is `Te1-Aux` suggesting that `DtRecon` improves the quality of $\mathcal{D}^{aux}$ to be similar to $\mathcal{D}_{tr}$. This helps train better shadow models for `MemInf`, `AttInf`, and `DistInf`. I validate the conjecture by comparing the attack accuracy of `MemInf`, `AttInf`, and `DistInf` in two settings: (i) when $\mathcal{D}^{aux}$ has no overlap with $\mathcal{D}_{tr}$, and (ii) when $\mathcal{D}^{aux}$ includes records extracted using `DtRecon`, and used to train the shadow models. I evaluate using the mean and standard deviation over five runs unless stated otherwise.

Experimental Setup: To validate the conjecture, I generate reconstructed data by executing `DtRecon` on a fraction of $\mathcal{D}_{tr}$. I then replace the fraction of $\mathcal{D}^{aux}$ with these reconstructed $\mathcal{D}_{tr}$ records (aka replacement ratio, I use {25%, 50%, 75%}). For `DtRecon`, I use generative gradient inversion [450]. I assume $\mathcal{A}dv1$ has whitebox access to $\mathcal{M}$ (e.g., in a federated learning setting where $\mathcal{A}dv$ can access the model gradients), and uses this to reconstruct data records in $\mathcal{D}_{tr}$. These reconstructed data records are then shared with $\mathcal{A}dv2$ to train shadow models for `MemInf`, `AttInf`, and `DistInf`.

For `MemInf`, I use the likelihood ratio attack [267] on CIFAR 10/100 (§5.3.1). Each dataset's 50k training images are split equally into a target pool (members) and $\mathcal{A}dv$'s non-member pool ($\mathcal{D}^{aux}$). $\mathcal{M}$ is trained on a random 12.5k subset of the target pool, while $\mathcal{A}dv$ trains 64 shadow models (50 epochs each) on random halves of a 15k-record $\mathcal{D}^{aux}$. $\mathcal{D}_{te}$ is the standard 10k records from the datasets. I evaluate on a fixed disjoint set of 2k records (1k members, 1k non-members). Overall, we have $|\mathcal{D}_{tr}| = 12500, |\mathcal{D}_{te}| = 10000, |\mathcal{D}^{aux}| = 25000$, and a membership evaluation set of size 2000. All models are ResNet34.

For `AttInf`, I use the attack from Aalmoes et al. [256] on CELEBA and UTKFACE (§5.3.2). For CELEBA, the target task is smile prediction and the sensitive attribute is sex; for UTKFACE, the target task is predicting sex and the sensitive attribute is race. Stratified on the label and the sensitive attribute, the target trains on $\mathcal{D}_{tr}$ (1000 members, of which $\mathcal{A}dv$ reconstructs 500 and 500 are held-out), $\mathcal{A}dv$ holds $\mathcal{D}^{aux}$ (1500 non-members), and evaluation queries the 500 held-out members and 500 held-out non-members. Overall, we have $|\mathcal{D}_{tr}| = 1000, |\mathcal{D}^{aux}| = 1500$, and an evaluation set of size 1000. Both $\mathcal{M}$ and attack model use ResNet18.

Finally, for `DistInf`, I use the blackbox attack of Suri et al. [121] on CELEBA and UTKFACE (§5.3.2), with sex as the sensitive attribute. $\mathcal{M}$ is trained at ratio α_1, and the attack asks whether $\mathcal{M}$'s ratio is α_1 or an alternative ratio α_2. I train 128 shadow models (64 with subgroup ratio α_1, 64 with α_2), each on 200 records for 12 epochs. Through each shadow model, I pass a fixed set of 500 query images (half male, half female), and record its loss on each which are used by a logistic-regression attack classifier to predict whether $\mathcal{M}$ was trained with α_1 or α_2. I report average accuracy over three runs. $\mathcal{M}$ and shadow models are ResNet18.

Results (`DtRecon` $\rightarrow$ `MemInf`): I report the TPR@0.01 FPR for different percentages of replaced records. For the first dataset, the baseline TPR of 1.50 increases to 2.50 at 25% replacement and further to 3.50 at 50%, before dropping to 2.10 at 75% replacement. Similarly, for the second dataset, the baseline TPR of 2.50 improves to 2.70 at 25% replacement and 2.80 at 50%, but decreases to 2.60 at 75% replacement. Overall, the TPR@0.01 FPR improves with moderate levels of replacement, while a higher replacement ratio of 75% leads to a decline in TPR for both datasets. I conjecture that the benefit of adding member-like records to the shadow training set saturates after a few hundred records, while each inexact reconstruction

acts as a slightly mislabeled member whose error accumulates as more are replaced. Thus, for moderate ratios, DtRecon can improve MemInf, confirming the conjecture.

Results (DtRecon → AttInf): I report the AUC score under ROC curve. For CELEBA, the baseline AUC at 0% replacement is 69.25% (average value), which increases to 71.45% at 25% replacement, 73.07% at 50%, and reaches 73.95% at 75% replacement. Similarly, for UTKFACE, the baseline AUC of 62.94% improves consistently with higher replacement ratios, increasing to 64.85% at 25%, 66.03% at 50%, and 67.42% at 75% replacement. I find that the improvement for 50% and 75% replacement gives significant performance improvement when compared to the baseline. Overall, for both datasets, increasing the percentage of replaced records leads to an improvement in AUC score. This confirms the conjecture that there is potential for collusion where DtRecon can help improve AttInf.

Results (DtRecon → DistInf): I report the attack accuracy results for DistInf under two ratio configurations: α_1=0.5 vs. α_2=0.1, and α_1=0.5 vs. α_2=0.9 (following results in Suri and Evans [12]). For UTKFACE with α_2=0.1, the baseline attack accuracy at 0% replacement is 62.66%, which increases to 74.06% at 25%, $\sim 76\%$ at 50% and 75% replacement. For α_2=0.9, the attack accuracy improves from a baseline of 60.31% to 72.50% at 25%, 72.81% at 50%, and further to 75.31% at 75% replacement.

For CELEBA with α_2=0.1, the baseline attack accuracy of 60.47% increases consistently to 64.53% at 25% replacement, 68.91 at 50%, and 71.09% at 75%. The strongest trend is observed for CELEBA with α_2=0.9, where the baseline accuracy of 63.44% increases substantially to 80.78% at 25% replacement, 82.81% at 50%, and reaches 87.66% at 75% replacement. Overall, across both datasets and ratio configurations, increasing the percentage of reconstructed records in $\mathcal{D}^{aux}$ consistently improves DistInf, supporting the conjecture that DtRecon can increase collusion potential.

Guideline Effectiveness: I evaluated the proposed guideline in two ways: (a) whether it explains prior work and predicts unexplored cases (Sections 5.2 and 5.3), and (b) whether its predictions can be empirically validated for unexplored cases of collusion (Section 5.3.1). For (a), I show that the guideline correctly explains all previously reported cases of potential collusion (Tables 5.4 and 5.5) and generates plausible conjectures for unexplored cases (Section 5.3). For (b), I validate the guideline's prediction in five unexplored cases of collusion. Together, they demonstrate that the guideline is effective in explaining and predicting collusion potential.

5.4 Accounting for Adversary's Characteristics

So far, I did not explicitly take $\mathcal{A}dv$'s characteristics into account but implicitly assumed that each $\mathcal{A}dv$ possesses the minimal set of characteristics needed to execute the attack. I now extend the framework to consider $\mathcal{A}dv$'s characteristics by discussing the impact of changing them on the collusion potential. I present Tables 5.8 and 5.9 showing the impact of collusion potential on varying $\mathcal{A}dv$'s characteristics. The notations for $\mathcal{A}dv$'s characteristics are based on the proposed streamlined threat model in Chapter 2, and summarized in Table 2.1. I consider the following two cases depending on which $\mathcal{A}dv$'s characteristics are changed:

Table 5.8: **Impact of $\mathcal{A}dv$'s characteristics on collusion potential ($\mathcal{A}dv$1 $\rightarrow$ Change; $\mathcal{A}dv$2 $\rightarrow$ Same):** Rows correspond to $\texttt{Att}_1$; we vary $\mathcal{A}dv$1's characteristics while $\mathcal{A}dv$2 is fixed; values range from weak (left) to strong (right). ★ $\rightarrow$ minimum requirement for $\mathcal{A}dv$1 to execute $\texttt{Att}_1$; ➤ $\rightarrow$ collusion potential with $\texttt{Att}_2$ increases by strengthening the characteristic from ★; empty cell $\rightarrow$ insufficient, unrealistic, or unnecessary. Refer to Chapter 2 for description of notations. Markings are based on prior work covering each of the attacks [6, 7, 8, 9, 10, 11].

$\texttt{Att}_1$	role	cap		opt		$\texttt{k}_\texttt{m}$							i				$\texttt{k}_\texttt{d}$					
		HbC	Mal	nonAdpt	Adpt	NoAccess	BBox	GBox	WBox	hardLbl	topK	fullPred	NoAccess	1Shot	KShot	Unlmtd	NoAccess	NoOvlap	PartOvlap	FullOvlap	Blind	NoBlind
Poisn	DtOwnr/DtProv		★	★		★							★						★	➤	★	➤
	ModTrnr		★	★	➤				★			★				★				★		★
	ModProv		★	★				★									★					
	CodeProv		★	★		★							★				★					
Evsn	Clnt		★	★	➤		★	➤	➤	★	➤	➤		★	➤	➤		★	➤	➤	★	➤
ModExt	Clnt	★		★	➤		★			★	➤	➤			★			★	➤	➤	★	➤
MemInf	Clnt	★		★	➤		★	➤	➤	★	➤	➤		★	➤	➤		★	➤	➤	★	➤
AttInf	Clnt	★		★	➤		★	➤	➤	★	➤	➤		★	➤	➤		★	➤	➤	★	➤
DistInf	Clnt	★		★	➤		★	➤	➤	★	➤	➤		★	➤	➤		★	➤	➤		★
DtRecon	Clnt	★		★	➤		★	➤	➤	★	➤	➤		★	➤	➤		★	➤	➤	★	➤

- $\mathcal{A}dv$**1** $\rightarrow$ **Change;** $\mathcal{A}dv$**2** $\rightarrow$ **Same**: Table 5.8 shows the minimum characteristics for each risk that I implicitly assumed so far (indicated as ★). Increasing $\mathcal{A}dv$1's characteristics will increase the effectiveness of $\texttt{Att}_1$ and the correlations with underlying factors (Tables 5.2 and 5.3). This, in turn, increases the potential for collusion and facilitates $\texttt{Att}_2$. From Table 5.8, I can determine whether the potential for collusion increases based on increasing

$Adv1$'s characteristics beyond the minimum requirements (indicated as ➤) when going from left (weak) to right (strong).

- Adv**1** $\rightarrow$ **Same;** Adv**2** $\rightarrow$ **Change**: Recall that the guideline identifies a potential for collusion when the *factors* required for $\texttt{Att}_2$ are satisfied by the *outcomes* of $\texttt{Att}_1$. Strengthening $Adv2$'s characteristics can increase the effectiveness of $\texttt{Att}_2$, though not necessarily by increasing collusion potential. In some cases, stronger characteristics may directly satisfy prerequisites required for $\texttt{Att}_2$, independent of $\texttt{Att}_1$'s outcomes, thereby reducing or eliminating the need for collusion. Therefore, collusion should be evaluated only over the factors for $\texttt{Att}_2$ that remain unmet after accounting for $Adv2$'s characteristics. Table 5.9 summarizes such cases, where ↓ indicates reduced collusion potential under a specific characteristic, and ✗ indicates that collusion with $\texttt{Att}_1$ becomes unnecessary. I illustrate how strengthening $Adv2$'s characteristics removes or reduces the need for collusion, using Table 5.9.

Table 5.9: **Impact of Adv's characteristics on collusion potential ($\texttt{Att}_2$):** Rows correspond to $\texttt{Att}_2$; I vary $Adv2$'s characteristics while $Adv1$ is fixed. Strengthening $Adv2$ raises effectiveness directly (not via collusion, hence omitted); I instead mark its effect on collusion potential with the listed $\texttt{Att}_1$ (s): ↓ $\rightarrow$ collusion potential decreases with characteristic; ✗ $\rightarrow$ collusion not required as characteristic meets the prerequisite factor. Columns under $\overline{\texttt{k}_\texttt{m}}$ are left empty for ModExt, as they are the outcomes of ModExt.

	$\overline{\texttt{k}_\texttt{d}}$			$\overline{\texttt{k}_\texttt{m}}$		
$\texttt{Att}_2$	$\texttt{Att}_1$	PartOvlap	FullOvlap	$\texttt{Att}_1$	GBox	WBox
Evsn	MemInf, DtRecon, AttInf, DistInf	↓	✗	ModExt	↓	✗
ModExt	MemInf, Evsn, DtRecon, AttInf, DistInf	↓	✗	—	—	—
MemInf	Evsn, DtRecon, AttInf, DistInf	↓	✗	ModExt	↓	✗
AttInf	MemInf, Evsn, DtRecon, DistInf	↓	✗	ModExt	↓	✗
DistInf	MemInf, Evsn, DtRecon, AttInf	↓	✗	ModExt	↓	✗
DtRecon	MemInf, Evsn, AttInf, DistInf	↓	✗	ModExt	↓	✗

  - **Model Knowledge:** Granting $Adv2$ whitebox access to $\mathcal{M}$ (WBox) satisfies both Te3-Shdw, Te4-Arch, which ModExt would otherwise provide. As a result, attacks such as MemInf, AttInf, DistInf, and DtRecon can be carried out effectively *without* colluding with $Adv1$ for ModExt (✗ in Table 5.9). Granting only graybox access (GBox) satisfies Te4-Arch but not Te3-Shdw; this *reduces* the need for collusion with $Adv1$ for ModExt (↓).

- **Data Knowledge ($\mathcal{D}^{aux}$):** MemInf, Evsn, and DtRecon produce $\mathcal{D}^{aux}$ as an outcome (Te1-Aux), which can strengthen other attacks. However, Te1-Aux may be satisfied directly by increasing $\mathcal{A}dv2$'s knowledge about $\mathcal{D}_{tr}$: full overlap (FullOvlap) removes the need to collude with $\mathcal{A}dv1$ for MemInf, Evsn, and DtRecon (✗), while partial overlap (PartOvlap) reduces it (↓).
- **Data Knowledge (distribution):** AttInf and DistInf provide Te2-Dist as an outcome, which strengthens attacks that require Te2-Dist, such as AttInf, Evsn, MemInf, DtRecon, and ModExt. Te2-Dist can instead be satisfied by increasing $\mathcal{A}dv2$'s knowledge about $\mathcal{D}_{tr}$: FullOvlap removes the need for collusion while PartOvlap reduces it.

Based on the above discussion, the guideline should be refined to account for the cases where strengthening $\mathcal{A}dv2$'s characteristics satisfies $\mathtt{Att}_2$'s prerequisites without requiring collusion. Specifically, after identifying common factors between $\mathtt{Att}_1$ and $\mathtt{Att}_2$ from Tables 5.2 and 5.3, I use Tables 5.8 and 5.9 to determine whether $\mathcal{A}dv2$'s characteristics already meet the prerequisites for $\mathtt{Att}_2$. Hence, when assessing collusion potential, I refine the guideline (G1) to consider only the common *unmet* factors between risks *after* accounting for $\mathcal{A}dv$'s characteristics.

Takeaway: I revise the guideline to account for the setting where $\mathcal{A}dv2$ has characteristics which meet the pre-requisite, removing the need for collusion.

5.5 Discussions

I discuss how the framework is *comprehensive* and *extensible* to new factors and attacks, how to extend the guideline beyond two colluding adversaries, and how to represent multiple colluding adversaries as a single entity.

Requirements Satisfied by Framework: I evaluate the framework against comprehensiveness (R1) and extensibility (R2).

***R1** Comprehensive:* The framework is comprehensive with respect to both the attacks considered for collusion and the factors used to relate them.

- *Comprehensiveness of Attacks:* The framework covers attacks corresponding to security, privacy, and fairness that are sufficiently studied in prior work to support systematic reasoning about collusion across the ML pipeline. These include both train-time and inference-time attacks for which the interactions can be characterized through common underlying factors. Attacks that lack sufficient prior work to enable such characterization were omitted:
  - *Side-channel attacks* assume BBox access, where Clnt or SrvProv exploits additional metadata (e.g., power consumption or timing differences) to extract $\mathcal{M}_{ar}$ and partially recover

θ [451, 452, 453].
  - *Fault injection* occurs when `Clnt` or `SrvProv` exploits hardware-level vulnerabilities to induce targeted faults in $\mathcal{M}$ execution or weight storage, enabling accuracy degradation or targeted misclassification without modifying the training data, model architecture, or inference logic [454, 455, 456].
  - *Adversarial initialization* is a train-time attack where `ModProv` or `ModTrnr` can initialize $\mathcal{M}$ such that standard training on benign data converges to $\mathcal{A}dv$-chosen behavior (e.g., backdoors, increased training time, or biased outputs) [457, 458].
  - *Data ordering attacks* are where `DtOwnr` or `DtProv` can reordering $\mathcal{D}_{tr}$ during training to introduce to $\mathcal{A}dv$-chosen behavior [459, 460].
  - *Sponge examples* violates availability of $\mathcal{M}$ where `Clnt` with `BBox` access can send carefully crafted queries to induce excessive energy consumption and latency [461, 462, 463].

  As research on these attacks matures, they can be incorporated into the framework by reasoning about $\mathcal{A}dv$'s characteristics and any additional factors required for the specific attacks.
- *Comprehensiveness of Factors:* The framework identifies a set of factors underlying both types of collusion, drawn from prior work and Chapter 2, Section 2.3. I show how *each attack* relates to *all* the identified factors, enabling systematic reasoning about collusion (Tables 5.2 and 5.3). While the factors may not be exhaustive, the framework is extensible and supports incorporating new ones (see discussion on extensibility in **R2**).
- *Comprehensiveness across Model Types/Dataset Modalities:* The proposed approach and identified factors are not limited to a specific model type; they also apply to generative models (e.g., language models). When explaining collusion in prior work, I successfully used the same factors to analyze generative models, suggesting their broader applicability. While additional generative-specific factors may exist, they were not included due to limited evidence of their correlation with attack success; however, such factors can be readily incorporated into the framework (see discussed on **R2** below).

Overall, the framework is comprehensive within the scope of attacks (meets **R1**).

***R2** Extensible:* The framework supports additional factors, attacks, and $\mathcal{A}dv$'s characteristics. To add new factors, one can specify how each factor relates to existing attacks, as illustrated in Tables 5.2 and 5.3. For new attacks, one can characterize how the attack relates to the existing factors (e.g., Tables 5.2 and 5.3 and Section 5.1.2). Once these are defined, the guideline can be used to study the potential collusion. Overall, the framework is extensible (meets **R2**).

Extending Beyond Two Adversaries: The framework extends to collusion with more than two adversaries. I first discuss two cases where $\mathcal{A}dv1$ colludes with multiple adversaries in *parallel* ("$\Rightarrow$" means "colludes with"):

- *Train-time $\mathcal{A}dv1 \Rightarrow$ Multiple inference-time adversaries*

$\mathcal{A}dv$1 can optimize train-time attacks to increase the effectiveness of multiple inference-time attacks, enabling collusion with several inference-time adversaries in parallel.

- *Inference-time $\mathcal{A}dv1 \Rightarrow$ Multiple inference-time adversaries*
  $\mathcal{A}dv$1 uses an inference-time attack and uses the knowledge learned as prerequisite to amplify the effectiveness of other inference-time attacks. This will allow $\mathcal{A}dv$1 to collude with multiple inference-time adversaries.

For the above two cases, the guideline can be applied for $\mathcal{A}dv$1 and each of the other inference-time adversaries in parallel. The potential for overall collusion will be determined independently by the potential for each pair of adversaries. I present a case for *sequential collusion*:

- *Inference-/Train-time $\mathcal{A}dv1 \Rightarrow$ Inference-time $\mathcal{A}dv2 \Rightarrow$ Inference-time $\mathcal{A}dv3 \Rightarrow \cdots$*
  $\mathcal{A}dv$1 colludes with $\mathcal{A}dv$2 to increase the effectiveness of an inference-time attack by $\mathcal{A}dv$2, whose success provides knowledge increasing the effectiveness of downstream inference-time attacks.

Here, I apply the guideline sequentially to each pair of adversaries. The potential for overall collusion will be determined by whether there is potential for collusion across all pairs of adversaries in sequence. More complex combinations of the parallel and sequential cases may exist, but the guideline can be readily extended to predict the potential for collusion.

Representing as Single $\mathcal{A}dv$: The two adversaries, $\mathcal{A}dv$1 and $\mathcal{A}dv$2, can be modeled as a single $\mathcal{A}dv$ when they share the same objective. For example, if $\mathcal{A}dv$1 performs poisoning specifically to increase the effectiveness of an inference-time attack executed by $\mathcal{A}dv$2, they can be represented jointly as a single $\mathcal{A}dv$. However, this abstraction overlooks scenarios in which $\mathcal{A}dv$1 and $\mathcal{A}dv$2 have different motivations and objectives. In such cases, $\mathcal{A}dv$1 may transfer attack outcomes or side information to $\mathcal{A}dv$2 (e.g., through a marketplace), enabling a more effective attack. Importantly, $\mathcal{A}dv$2 may only require a subset of the information obtained through $\texttt{Att}_1$, rather than the complete attack output.

Exceptions to Guideline: I do not claim the framework is perfect; rather, it is a first step toward systematically understanding and predicting the potential for collusion among adversaries. The guideline cannot assess collusion when no common factors exist. Further work is needed to either uncover additional relationships between attacks and factors or identify missing factors that may enable collusion. This would then allow the guideline to be applied to evaluate collusion among adversaries.

Adaptive Strategies: One possible setting involves multi-round collusion, where $\mathcal{A}dv$1 iteratively adapts their attack strategy based on the success of $\mathcal{A}dv$2's attack in the previous round. In this scenario, the effectiveness and collusion potential are expected to increase across rounds. Specifically, improvements in $\texttt{Att}_1$ by $\mathcal{A}dv$1 based on feedback from previous round, will further improve the effectiveness of $\texttt{Att}_2$ executed by $\mathcal{A}dv$2. This iterative amplification

is captured by the guideline through an increase in correlation between $\texttt{Att}_1$ and the factors across successive rounds, consequently explaining the increase in effectiveness of $\texttt{Att}_2$.

Scope of Attacks: The scope of attacks explored in this work is limited to well-studied security, privacy, and fairness risks with sufficient prior literature to establish correlations and validate the guideline. Additional attacks are possible, as discussed under "Framework and Requirements: **R1** (Comprehensive)". However, further empirical analysis is required to establish the corresponding correlations and apply the guideline.

Application of Framework and Mitigations: Various stakeholders in the ML pipeline can be impacted by the increased effectiveness of colluding adversaries. For instance, *data owners*, *model owners*, and *service providers* become more susceptible to security, privacy, and fairness risks under collusion. The framework helps identify possible sources of collusion, the enabling factors, and the potential impact of such collusion. *ML security researchers and practitioners* can use the framework to study previously unexplored collusion settings, develop stronger attacks for auditing, design defenses, and analyze enabling factors. Practitioners can further account for these factors to reduce collusion potential (see below), and use the guideline to efficiently assess collusion risks.

By explicitly identifying the enabling factors through the framework and guideline, the work can help researchers and practitioners design defenses that disrupt collusion. For train-inference collusion, defenses such as detecting and filtering poisoned samples in $\mathcal{D}_{tr}$ [464], modifying training objectives to reduce the influence of poisons [465], can reduce the impact of $\texttt{Poisn}$ that enable subsequent attacks. Additionally, specific factors can be explicitly targeted to reduce correlations (e.g., reduce subgroup distinguishability using fairness constraints [256]). For inference-time collusion, defenders can apply various transformations or filters on inputs and outputs to restrict the information that $\mathcal{A}dv$ can infer through interactions with $\mathcal{M}$. More broadly, combining complementary defenses that mitigate different risks can further reduce collusion potential (Chapter 4).

Contribution in Support of the Thesis Statement

Existing work lacks a framework to study the potential for collusion among adversaries where executing an attack can amplify the effectiveness of others. I present the *first framework* by identifying factors enabling *train-inference collusion* and *inference-time collusion*. For both collusion types, I observe that a successful attack results in changing some factors as an outcome, which can influence the effectiveness of other attacks relying on those factors. The proposed framework identifies such enabling factors to explain prior work and conjecture about unexplored collusion, thus, answering **RQ1.3** to support the thesis statement.

Part 2
Accountability in Machine Learning Pipelines

We need technical mechanisms to validate practitioners' claims that they have deployed specific defenses against risks and that they have evaluated and addressed the unintended interactions identified in **Part-1**, before deployment. Currently, practitioners can make claims about various ML properties using *ML property cards* (e.g., *datasheets*, *model cards*, and *inference cards*). However, these property cards are not verifiable, and a malicious practitioner may include false or misleading information. To address this limitation and enable verifiable ML property cards, I explore technical mechanisms that allow practitioners to demonstrate properties of an *ML model*, its *training process*, and its *training data* to a verifier (e.g., a regulator or customer). Such mechanisms can strengthen accountability in the ML pipeline and support regulatory compliance. In **Part-2**, I defend the following:

Thesis Statement

Ensuring accountability of ML operations by designing technical mechanisms that are effective, efficient, versatile, scalable, and robust, is feasible.

I first present background on the relevant concepts (Chapter 6), and then explore the design of technical mechanisms using both software-based and hardware-based approaches in separate chapters, each addressing the corresponding research questions to help support the thesis statement: **(RQ2.1)** *Can software-based ML property attestation mechanisms meet all requirements?* (Chapter 7); and **(RQ2.2)** *Can hardware-based ML property attestation mechanisms meet all requirements?* (Chapter 8).

Chapter 6

Background

I present an overview of datasheets and model cards (Section 6.1), cryptographic primitives including secure multi-party computation and zero-knowledge proofs (Section 6.2), and trusted execution environments (Section 6.3).

6.1 Datasheets and Model Cards

To improve transparency into ML training and inference, industry (e.g., Hugging Face and Google) has adopted *datasheets* and *model cards* to describe various properties about training datasets and models (i.e., training process and evaluation).

Datasheets (or *dataset cards*) [64, 466] describe properties of the training dataset ($\mathcal{D}_{tr}$) such as the collection process and data sources, pre-processing techniques used on the raw data (e.g., cleaning and labeling), intended use, composition of the dataset, and its distribution. They have been used extensively by companies such as Hugging Face [467].

Model cards describe various model properties [65], to provide better transparency into ML model design (including training and evaluation), prior to deployment for inference. They contain a variety of information including: (i) *general properties* (e.g., the model's intended use, risks, and other limitations); (ii) *training properties* (e.g., $\mathcal{D}_{tr}$ used, training configuraation $\mathcal{T}$ and model architecture $\mathcal{M}_{ar}$); and (iii) *evaluation properties* (e.g., various metrics such as accuracy, robustness, privacy, and fairness; and $\mathcal{D}_{te}$ used). They have been adopted by various companies such as Google [468] and Hugging Face [469].

In the same vein as model cards and datasheets, I introduce **inference cards** to describe the properties of each inference. For instance, a service provider may wish to demonstrate that

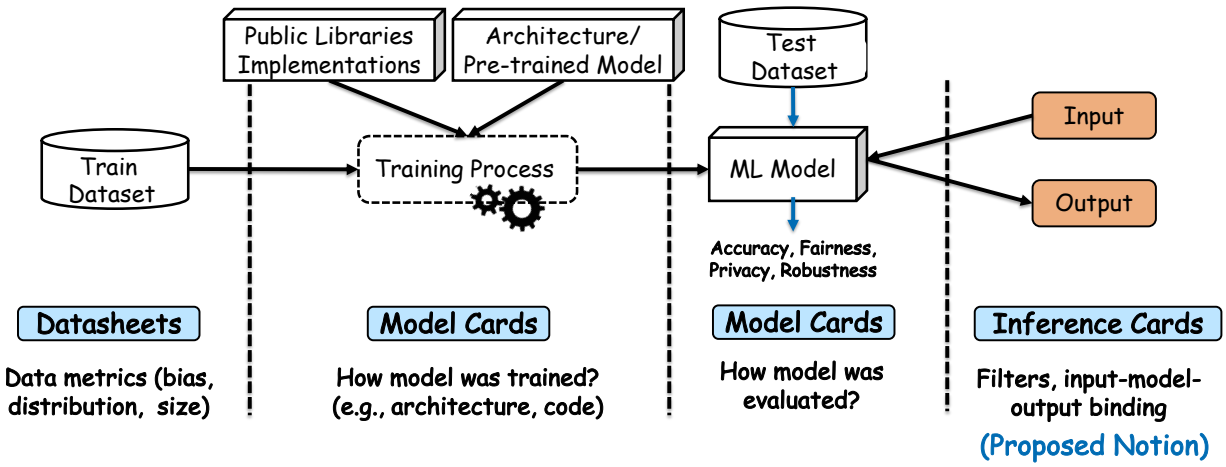


Figure 6.1: **Overview of ML property cards:** Datasheets describe properties of $\mathcal{D}_{tr}$ (e.g., bias and attribute distribution), model cards document training and evaluation details (e.g., how and on what the model was trained), and inference cards capture inference-time behavior (e.g., input–model–output relationships and applied filters).

a given output was produced by a specific model for a particular input, or that a corresponding model explanation is derived from the same inference [57]. Prior work have used non-interactive ZKPs for verifiable inference in ML classifiers [470, 471, 472] and LLMs [473], to provide functionality similar to inference cards, but with verifiability.

Collectively, I refer to them as *ML property cards*, which contain information about *ML properties* such as intended use, risks and limitations, training process, datasets, and metrics for evaluation (e.g., accuracy, fairness, robustness) [65]. Figure 6.1 summarizes ML property cards across the ML pipeline: *pre-training* (datasheets for $\mathcal{D}_{tr}$'s properties), *in-training* (model cards for training process), and *post-training* (model cards for evaluation and inference cards).

6.2 Cryptographic Primitives

I describe zero-knowledge proofs (ZKPs), and multi-party computation (MPC).

Zero-Knowledge Proofs

A zero-knowledge proof (ZKP) is a way for the prover ($\mathcal{P}rv$) to convince a verifier ($\mathcal{V}er$) that a statement is true without revealing anything beyond the truth of the statement. $\mathcal{P}rv$ has some

secret evidence (called a witness) that proves the statement. The goal is to convince $\mathcal{V}er$ that $\mathcal{P}rv$ genuinely possesses this evidence without actually showing it. A valid zero-knowledge proof must satisfy three key properties:

- **Completeness**: If the statement is true and $\mathcal{P}rv$ is honest, $\mathcal{V}er$ is convinced.
- **Soundness**: If the statement is false, no cheating $\mathcal{P}rv$ can trick $\mathcal{V}er$ into believing it (except with extremely small probability).
- **Zero-knowledge**: $\mathcal{V}er$ learns only that the statement is true; anything it sees could be simulated without access to $\mathcal{P}rv$'s secret.

Multi-party Computation

Secure multi-party computation (MPC) enables a set of parties to jointly compute a function over their private inputs while revealing nothing beyond the outputs [474]. MPC has been adopted for a wide range of applications, including financial services [475] and privacy-preserving ML [476]. Let n parties $Prt_1, \dots, Prt_n$ hold private inputs $x_1, \dots, x_n$ to jointly compute a function $f(x_1, \dots, x_n) \rightarrow (y_1, \dots, y_m)$ where y_i is the output delivered to party Prt_i. An MPC protocol Π specifies the interaction among the parties to compute f.

- **Correctness:** Protocol Π is correct if, for all inputs $(x_1, \dots, x_n)$,
$$\mathbb{P}[\Pi(x_1, \dots, x_n) = f(x_1, \dots, x_n)] = 1,$$
up to negligible error for probabilistic protocols.
- **Security (simulation-based):** Let $\mathcal{A}dv$ corrupt some of the parties. A protocol is considered secure if, for every real-world $\mathcal{A}dv$, there exists a simulator such that $\mathcal{A}dv$'s view during the real execution is computationally indistinguishable from what the simulator can generate using only the inputs of the corrupted parties and the final output of the function. Computationally indistinguishable means $\mathcal{A}dv$ cannot tell apart the real execution from the simulated one, and learns nothing more than their inputs and the output.

There are two threat models commonly considered in MPC protocols:

- In the *semi-honest or honest-but-curious threat model*, parties follow the protocol correctly but may try to learn additional information from the messages they receive [477, 478].
- In the *malicious threat model*, parties can deviate arbitrarily from the protocol, requiring stronger security guarantees such as verifiability and consistency checks [479].

While maliciously secure MPC protocols provide stronger security in malicious threat model, they come with higher computation and communication costs [479]. For real-world applications, semi-honest security guarantees are often sufficient and provide baseline performance benchmarks [477, 478].

Common techniques for building MPC protocols include (i) *secret sharing*, where each input is split into multiple shares so that no small subset of shares reveals the original input, and (ii) *garbled circuits*, where the function to be computed is encoded as an encrypted Boolean circuit that can be evaluated without exposing the underlying inputs.

In this thesis, I will focus on secure two-party computation (2PC) using secret sharing with one dishonest party (can be either semi-honest or malicious). Unlike garbled circuits, secret sharing-based protocols are especially efficient for arithmetic-heavy computations (e.g., additions and multiplications over fields), and many operations (like addition) are essentially "free" (i.e., no communication cost). This is specially useful for ML-based workloads.

6.3 Trusted Execution Environments

Trusted execution environments (TEE) [480] provide an environment in which security-critical code (trusted applications or TAs) can run without outside interference from other TAs, or the rest of the system (Rich Execution Environment or REE, which contains the operating system and normal software). TEEs can be described in terms of three main properties: *isolation*, *protected memory*, and *remote attestation* [480].

- **Isolation** refers to the inability of outside software to interfere with the operation of a TA, or to read its confidential state. TEEs provide strong isolation between trusted and untrusted components. Code and data inside a TEE are protected from external software, including the operating system and hypervisor. Access is limited to well-defined interfaces, keeping execution confidential and resistant to tampering.
- **Protected memory** refers to persistent memory made available to TAs that resists interference by other software. This memory is encrypted and integrity-protected, which ensures that sensitive data cannot be read or modified by untrusted components. Any attempt to tamper with memory contents is detected, preserving both data confidentiality and integrity.
- **Remote attestation** allows $\mathcal{P}rv$ to convince $\mathcal{V}er$ that a specific program is running inside a genuine TEE. The TEE computes a measurement and produces an attestation quote $\sigma = \text{Sign}_{sk_{\text{TEE}}}(\text{Hash}(Code), \text{nonce})$, where sk_{TEE} is a hardware-protected secret attestation key for signing. $\mathcal{V}er$ checks if $\text{Verify}_{pk_{\text{TEE}}}(\sigma, \text{Hash}(Code), \text{nonce}) = 1$, where pk_{TEE} is the public key. This ensures authenticity (genuine TEE) and integrity (correct code is running).

Property Attestation: Remote attestation can be extended to prove high-level properties of the executed code or system state (aka property attestations) [481, 482]. This includes attesting whether $\mathcal{P}rv$'s system satisfies the desired (security) requirements without revealing its specific software or hardware configuration. For a property of interest P that can be extracted

from the code (e.g., software or hardware configuration), $\mathcal{P}rv$ generates an attestation over P given by $\sigma = \text{Sign}_{sk_{\text{TEE}}}(\text{P}, \text{nonce})$. $\mathcal{V}er$ checks if $\text{Verify}_{pk_{\text{TEE}}}(\sigma, \text{P}, \text{nonce}) = 1$. This allows $\mathcal{V}er$ to only learn that the system satisfies P, without requiring full disclosure of the code, enabling confidentiality and verifiability.

Intel's Software Guard Extensions (SGX): I focus on Intel's SGX [483], though any TEE technology that offers isolation and remote attestation is applicable. SGX provides TEEs that allow applications to create an *enclave* in their own memory space and run a TA of their choice. SGX ensures that neither the host application, the operating system, nor external code can access the enclave's memory, with communication occurring through defined entry/exit points and via memory read/write operations between the enclave and the host application.

SGX enclaves can remotely attest themselves by obtaining a *quote*, which serves as a signature over TA-provided data and certifies the enclave's identity. This identity includes `MRENCLAVE` (a hash of the TA's initial memory), `MRSIGNER` (a hash of the public key that signed the TA), user data (a chunk of data provided by the TA), versioning, and other data [483, EREPORT].

$\mathcal{V}er$ can assess the validity of a quote, then use the hash stored in its `MRENCLAVE` field and knowledge of the corresponding code's behavior to determine the TA's state when the quote was generated. If a TA produces quotes only for specific computations, $\mathcal{V}er$ can confirm that the computation was performed, and the quoted result was produced. This enables proving properties of data and computations within the TA.

Gramine Library OS: Gramine[1] can run unmodified applications inside an SGX enclave using a library OS, enabling the use of off-the-shelf software such as Python. This requires a manifest specifying the application's security configuration, including which files are accessible and whether integrity checks apply. By default, only explicitly listed *trusted files* (hashed at build time and integrity-checked) and *allowed files* (accessible without integrity checks) can be accessed within the enclave. Gramine automatically hashes trusted files and adds them into the enclave's `MRENCLAVE`. As a result, trusted files cannot be replaced by a malicious host without detection, allowing security-critical files to be stored outside the enclave.

[1] https://gramineproject.io/

Chapter 7

Software-based ML Property Attestation Mechanisms

Overview

To prevent practitioners from making false claims in ML property cards, these cards must be verifiable. To this end, I introduce *ML property attestation mechanisms* that enable a prover (e.g., a model trainer) to demonstrate ML properties to $\mathcal{V}er$ (e.g., a regulator) without revealing the model or training data. An ideal mechanism should be *effective, efficient, versatile, scalable, and robust*. Using distributional equity as a case study, I explore two software-based mechanisms: ML-based and cryptographic (via multi-party computation). I also propose a hybrid mechanism that improves efficiency over cryptographic mechanism, and effectiveness over ML-based one. However, it does not scale to multiple verifiers (negatively answering **RQ2.1**), suggesting the need for other mechanisms to support the thesis statement.

Regulations strive to ensure that ML properties, including the training process and model behavior during evaluation and inference, are as expected [484, 485, 486, 487, 488, 489, 490, 491]. For instance, a model provider may need to convince a verifier $\mathcal{V}er$ (e.g., a regulator or potential customer purchasing a model $\mathcal{M}$) that $\mathcal{M}$ meets required standards for accuracy, fairness, privacy, or robustness [484, 491, 490, 489, 488]. They may also need to demonstrate that the distributional properties of the training data comply with relevant regulations [486, 487]. Privacy or confidentiality considerations may imply that practitioners need to demonstrate regulatory compliance *without disclosing the model or the training data to* $\mathcal{V}er$. Thus, for better

transparency, practitioners[1] need mechanisms to demonstrate ML properties.

To this end, ML property cards can be used which include (i) *datasheets* to show training data properties (e.g., bias and distribution) [64, 466], (ii) *model cards* to show how $\mathcal{M}$ was trained and evaluated (e.g., accuracy, fairness, privacy, and robustness) [65], and (iii) *inference cards* to capture properties during inference (e.g., binding an output to $\mathcal{M}$ and an input). Since these are provided by practitioners, verifiers must trust their claims without independent verification. However, malicious practitioners may falsify information in ML property cards (e.g., model and dataset descriptions on Hugging Face can be inaccurate [66]).

To ensure that the information in *ML property cards* is *neither false nor tampered with*, we need technical mechanisms to provide *verifiable claims* about ML properties. I introduce *ML property attestation mechanisms* that allow a *prover* $\mathcal{P}rv$ (e.g., a model trainer) to demonstrate relevant ML properties to $\mathcal{V}er$. The attestations from these mechanisms can be combined with ML property cards to make them *verifiable*, thus enabling accountability of the practitioners' claims and the ML operations. Such mechanisms can also help with regulatory compliance by comparing the attested information against expected behavior.

Ideally, an ML property attestation mechanism should be *effective* (accurately measure properties), *efficient* (incur minimal overhead), *versatile* (cover diverse properties), *scalable* (validated by multiple verifiers), and *robust* (resistant to evasion by a malicious $\mathcal{P}rv$). A natural research questions is: (**RQ2.1**) *can software-based ML property attestation mechanisms meet all requirements?* Software-based mechanisms include re-purposing (privacy) inference attacks to attest ML properties (e.g., using distribution inference to infer the characteristics of the training dataset), or using cryptographic primitives (e.g., 2PC and ZKPs). Here, $\mathcal{V}er$ runs such protocols on $\mathcal{P}rv$'s $\mathcal{M}$.

I answer **RQ2.1** using the case study of *distributional equity* which uses proof of attribute distribution, to check whether the training dataset's *distributional properties* with respect to specific attributes (e.g., sex or race), are consistent with the proportion in the population at large (i.e., meets some required property). This ensures that there are no errors from population misalignment, i.e., $\mathcal{M}$ does not accurately represent the target population due to distribution shifts between the training dataset and inference-time data [492]. Additionally, regulations require model providers to demonstrate such *distributional equity* in their training dataset, showing that the distributional properties of certain attributes fall within the ranges specified by regulatory requirements:

- *Algorithmic Accountability Act* bill [487] requires the operators of automated decision sys-

[1]**Terminology:** I use *practitioners* broadly to describe model designers, trainers, and service provers who design, train, and deploy ML models for various applications.

tems to keep track of "the representativeness of the dataset and how this factor was measured including ...the distribution of the population" (cf. [487, §7.C.(i)]).
- EU's AI act [488] stipulates that "datasets ...shall have the appropriate statistical properties, including, where applicable, as regards the persons or groups of persons on which the high-risk AI system is intended to be used" (cf. [488, Art. 10.3]).

I claim the following contributions:

1. the novel notion of *ML property attestation*, and a desiderata for an ideal ML property attestation mechanism; (Section 7.1)
2. using proofs of attribute distribution, I evaluate two software-based ML property attestation methods (ML-based and cryptographic) and propose a hybrid that is more effective than ML-based mechanism, and more efficient than cryptographic mechanism[2]; (Section 7.2)
3. a comprehensive empirical evaluation of different mechanisms against the ideal requirements to identify their benefits and limitations. (Sections 7.3 and 7.4)

7.1 ML Property Attestation

The goal is to identify the requirements for an ideal ML property attestation mechanism, and explore the design space of software-based mechanisms to see if they meet all the requirements. I assume $\mathcal{P}rv$ trains, evaluates, or deploys $\mathcal{M}$. The various ML property attestations across the ML pipeline include the following:

- ***Proof*[3] *of Attribute Distribution***: $\mathcal{P}rv$ can attest to the proportion of training records with a given attribute value (e.g., the fraction where sex is male). Formally, this shows that its training dataset $\mathcal{D}_{tr}^{prv}$ satisfies the property P.
- ***Proof of Training***: $\mathcal{P}rv$ can attest that $\mathcal{M}$ was trained using a specific training configuration $\mathcal{T}$, model architecture $\mathcal{M}_{ar}$, and training dataset $\mathcal{D}_{tr}^{prv}$.
- ***Proof of Evaluation***: $\mathcal{P}rv$ evaluates $\mathcal{M}$ to show that it satisfies some accuracy, fairness and robustness requirements prior to deployment.
- ***Proof of Inference***: Once $\mathcal{P}rv$ deploys $\mathcal{M}$ as a service, clients can query $\mathcal{M}$ by sending an input $\mathcal{I}$ via an API interface and obtain the corresponding output $\mathcal{O}$. Here, $\mathcal{P}rv$ attests that $\mathcal{O}$ was obtained from $\mathcal{M}$ for a specific $\mathcal{I}$.

$\mathcal{V}er$ does not trust $\mathcal{P}rv$ who can subvert the attestation mechanism to fool $\mathcal{V}er$ into believing that some property holds when it does not. For instance, $\mathcal{P}rv$ can make false claims despite

[2]Code: https://github.com/ssg-research/distribution-attestation.
[3]**Terminology:** I use "proofs" to indicate specific examples of attestations for ML properties.

not using the expected $\mathcal{T}$ or $\mathcal{M}_{ar}$, using a different dataset than $\mathcal{D}_{tr}^{prv}$, which does not satisfy the property P, incorrect values for various metrics (e.g., accuracy, fairness, and robustness), or using a different $\mathcal{O}$ for inference. To prevent a malicious prover ($\mathcal{P}rv_{mal}$) from making false claims, ML property attestations can be combined with ML property cards to make them *verifiable*. Then, $\mathcal{V}er$ can check for the integrity of the content in the property cards with the help of the attestations.

Requirements for ML Property Attestation Mechanisms: An ideal ML property attestation mechanism must be:

R1 Effective (correctly estimates ML properties of interest)[4];
R2 Efficient (incurs an acceptable overhead);
R3 Scalable (can be efficiently checked by multiple verifiers);
R4 Versatile (supports a wide variety of ML properties and new properties); and
R5 Robust (resists evasion by a malicious prover $\mathcal{P}rv_{mal}$).

Software-based ML Property Attestation Mechanisms: I discuss two software-based attestation mechanisms: ML-based and cryptographic (includes ZKPs and MPC).

- **ML-based mechanisms** include statistical techniques or re-purposing various privacy inference attacks for attesting $\mathcal{M}$. For instance, membership inference attacks can be used to check for compliance with "Right to Erasure" [493, 494, 495, 496]. Similarly, distribution inference attacks can be used to check for distributional properties of $\mathcal{D}_{tr}$ (see Section 7.2). "Proof of Learning" (or proof of training) proves that $\mathcal{M}$ was trained on a specific dataset [497].
- **Cryptographic mechanisms** include two primitives for attesting ML properties:
  - ZKPs have been widely used for verifiable inference to ensure model outputs are untampered [470, 472, 498, 471, 499, 500]. ZKPs for training, though in early stages, have been used for proof of training [501, 502, 503, 504, 505]. ZKPs can also verify metrics during evaluation such as fairness [502, 506, 507, 508] and privacy [509].
  - MPC has advanced significantly for private training and inference [510, 511, 512, 513, 514], leading to broader adoption in ML. It has also been applied to verify end-to-end training and inference pipelines, though at the cost of multiple interactions [515].

[4]**Terminology:** "Effectiveness" has a distinction between cryptographic and ML-based mechanisms. Even if the mechanisms are correctly designed and implemented, ML-based attestation is inherently probabilistic and may produce errors making it ineffective. By contrast, for cryptographic mechanisms, effectiveness is from the implementation correctness to measure the required ML properties.

7.2 Case Study: Distributional Equity

To evaluate whether various software-based mechanisms meet all requirements (R1-R5), I use the case study of *distributional equity*. Here, $\mathcal{P}rv$ uses *proof of attribute distribution* to show the training dataset's *distributional properties* with respect to specific attributes (e.g., sex or race). $\mathcal{V}er$ then checks if these properties are consistent with some required property p_{req} (determined by company policies or regulation).

I define distributional property as the ratio of an indicator function, counting different data records applied to a dataset (uniformly sampled from a distribution) with a specific attribute value (e.g., males), and the total number of data records [12]. Examples include the ratio of males to females or whites to non-whites in tabular or image datasets, or the average node degree and clustering coefficient for graph data [12, 121].

System and Adversary Models: I assume that the distributional property for the attribute of interest can take a set of n possible values $\overline{\mathbf{p}}$ ={p_0,...,p_n} (e.g., proportion of females in $\mathcal{D}_{tr}^{prv}$). The goal of $\mathcal{P}rv$, who has trained a model $\mathcal{M}$ on $\mathcal{D}_{tr}^{prv}$, is to succeed in ML property attestation to comply with regulations. $\mathcal{V}er$'s goal is to ensure that attestation succeeds if $\mathcal{D}_{tr}^{prv}$ meets p_{req} even if $\mathcal{P}rv$ tries to evade the attestation process. Here, $\mathcal{P}rv$ is malicious and can attempt to evade the attestation while $\mathcal{V}er$ is trusted. However, $\mathcal{P}rv$ does not want to disclose $\mathcal{D}_{tr}^{prv}$ to $\mathcal{V}er$ for confidentiality, but has given $\mathcal{V}er$ whitebox access to $\mathcal{M}$ (required for ML-based attestation and clarified below under "Mechanism 1: ML-based").

Revisiting Requirements: I focus on empirically evaluating three requirements: effective (R1), efficient (R2), and robust (R5), and discuss the remaining two requirements: scalable (R3) and versatile (R4), towards the end of Section 7.4. For effectiveness (R1), the mechanism should correctly identify whether $\mathcal{D}_{tr}^{prv}$ meets p_{req}, with acceptably low false accepts (FA) or false rejects (FR). For robustness (R5), the mechanism should be effective(R1), with respect to FA, even if $\mathcal{P}rv$ misbehaves.

I present three ML property attestation mechanisms: ML-based mechanism, cryptographic mechanism using MPC, and a hybrid mechanism combining the benefits of both.

Mechanism 1: ML-based

Recall that distribution inference attacks infer statistical properties of training data given access to the target model. These attacks can be re-purposed for ML property attestation. However, directly applying distribution inference attacks is not possible as there are differences between the two settings (inference attack vs. attestation) in terms of their:

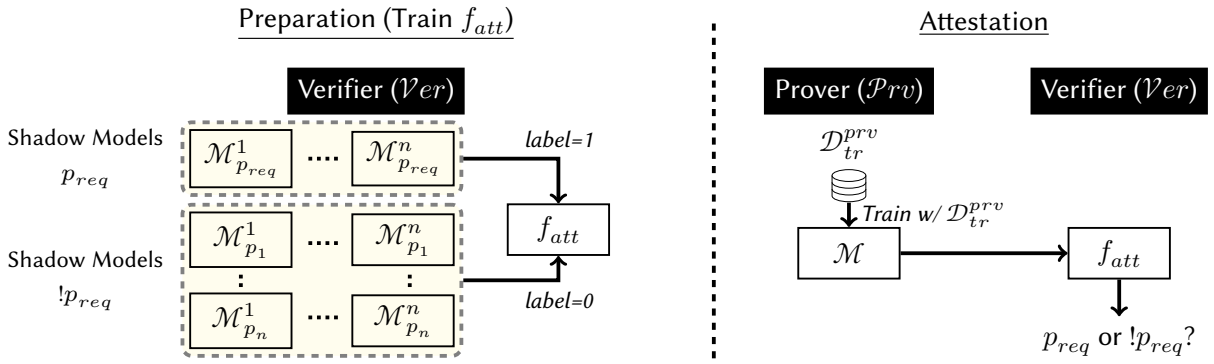


Figure 7.1: **Inference-based attestation**: During preparation, $\mathcal{V}er$ trains f_{att} using the first layer parameters of models trained on $\mathcal{D}_{tr}^{prv}$ with p_{req} ($\{\mathcal{M}^i_{p_{req}}\}_{i=1}^{\mathcal{N}}$) and $!p_{req}$ ($\{\mathcal{M}^i_{!p_{req}}\}_{i=1}^{\mathcal{N}}$). During attestation, $\mathcal{V}er$ uses the first layer parameters of $\mathcal{M}$ to attest whether it was indeed trained on $\mathcal{D}_{tr}^{prv}$ with p_{req} or not.

- **Objective**: Attack distinguishes between two property values while attestation requires differentiating p_{req} from all others ($!p_{req}$).
- **Requirement**: Attestation has the additional robustness requirement (**R5**) to resist $\mathcal{P}rv_{mal}$'s attempts to fool $\mathcal{V}er$.

I show how distribution inference attacks can be adapted for attestation, and describe the ML-based attestation below.

Assumptions: Unlike the attack where whitebox model access for $\mathcal{A}dv$ is a strong assumption, I assume a setting where $\mathcal{P}rv$ is willing to share $\mathcal{M}$ but not $\mathcal{D}_{tr}$, and where both $\mathcal{P}rv$ and $\mathcal{V}er$ are incentivized to collaborate to complete the attestation successfully[5].

Setup: Following the literature on distribution inference attacks, I assume that both $\mathcal{P}rv$ and $\mathcal{V}er$ know some underlying data distribution $\mathbb{D}_{\mathsf{P}}$ which satisfies property P [12, 449, 120, 121]. $\mathcal{P}rv$'s data $\mathcal{D}^{\mathcal{P}rv} \sim \mathbb{D}_{\mathsf{P}}$ is split into $\mathcal{D}_{tr}^{prv}$, $\mathcal{D}_{te}^{prv}$, and $\mathcal{D}_{\mathsf{P}}^{vrf}$. $\mathcal{D}_{tr}^{prv}$ is used to train $\mathcal{M}$ with property P, and $\mathcal{D}_{te}^{prv}$ is used to evaluate $\mathcal{M}$ using various metrics (e.g., accuracy, fairness, robustness). $\mathcal{V}er$'s dataset $\mathcal{D}^{\mathcal{V}er} \sim \mathbb{D}_{\mathsf{P}}$ is split into a training dataset ($\mathcal{D}_{tr}^{\mathcal{V}er}$) for designing the attestation mechanism, and a test dataset ($\mathcal{D}_{te}^{\mathcal{V}er}$) to locally evaluate the attestation mechanism (by $\mathcal{V}er$). Once the attestation mechanism is finalized, the final evaluation is done using $\mathcal{D}_{\mathsf{P}}^{vrf}$.

[5]It is possible that $\mathcal{P}rv$ is unwilling to share $\mathcal{M}$ with $\mathcal{V}er$ due to confidentiality or intellectual property concerns. In such cases, blackbox distribution inference attacks can be re-purposed as an attestation mechanism. However, to establish an upper bound on their effectiveness, I instead consider the stronger whitebox setting, where such attacks are more effective [12]. Thus, I assume the setting where $\mathcal{P}rv$ is willing to share $\mathcal{M}$ with $\mathcal{V}er$.

Method: Given access to $\mathcal{M}$, $\mathcal{V}er$ uses an attestation classifier (f_{atst}) to attest whether $\mathcal{D}_{tr}^{prv}$ satisfies p_{req} using the first layer parameters of $\mathcal{M}$ as input to f_{atst}. The first layer parameters are more effective to capture distributional properties for successful distribution inference than subsequent layers [12]. To train f_{atst}, $\mathcal{V}er$ uses $\mathcal{D}_{tr}^{\mathcal{V}er}$ and generates multiple sub-distributions $\{\mathcal{G}_0(\mathbb{D}_{\mathsf{P}}), \ldots, \mathcal{G}_n(\mathbb{D}_{\mathsf{P}})\}$ corresponding to property values in $\overline{\mathbf{p}}$ and samples datasets $\{\mathcal{D}_0, \ldots, \mathcal{D}_n\}$. In practice, this is done by sampling datasets multiple times with different properties from $\mathcal{D}^{\mathcal{V}er}$. For each dataset and property value, $\mathcal{V}er$ trains $\mathcal{N}$ "shadow models" $\{\{\mathcal{M}_0^i\}_{i=1}^{\mathcal{N}}, \ldots, \{\mathcal{M}_n^i\}_{i=1}^{\mathcal{N}}\}$. These mimic the models that $\mathcal{V}er$ could encounter during attestation. $\mathcal{V}er$ trains f_{atst} using the first layer model parameters trained on $\mathcal{D}_{tr}^{prv}$ with p_{req} ($\{\mathcal{M}_{p_{req}}^i\}_{i=1}^{\mathcal{N}}$) and $!p_{req}$ ($\{\mathcal{M}_{!p_{req}}^i\}_{i=1}^{\mathcal{N}}$). $\mathcal{V}er$ uses $\mathcal{D}_{te}^{\mathcal{V}er}$ for evaluating f_{atst}. Once f_{atst} is finalized by $\mathcal{V}er$, the evaluation of the attestation mechanism is done using $\mathcal{D}_{\mathsf{P}}^{vrf}$. I present a visualization of the ML-based mechanism in Figure 7.1.

Mechanism 2: Cryptographic

ML property attestation can be securely achieved using cryptographic protocols (e.g., MPC, ZKPs) by proving that (a) $\mathcal{D}_{tr}^{prv}$ meets p_{req} (`DistCheck`), and (b) $\mathcal{M}$ was trained on $\mathcal{D}_{tr}^{prv}$ to ensure that a misbehaving $\mathcal{P}rv$ does not change $\mathcal{D}_{tr}^{prv}$ after (a) (Figure 7.2). Generating attestations using ZKPs for each training epoch is inefficient, limiting them to simple models [502, 503, 504]. Extending them to neural networks is an open problem due to their significant overhead [501, 503, 504]. Compared to ZKPs, MPC has been developed with readily available frameworks for ML. Hence, I focus on MPC and specifically 2PC in the rest of the text.

Assumptions: $\mathcal{P}rv$ may deceive $\mathcal{V}er$ about p_{req}, acting maliciously. However, $\mathcal{V}er$, interested in purchasing $\mathcal{M}$, has no incentive to cheat, but may seek additional details about $\mathcal{D}_{tr}^{prv}$. Thus, I assume $\mathcal{V}er$ behaves semi-honestly. Note that *whitebox access is not needed* for cryptographic mechanism: $\mathcal{V}er$ never sees $\mathcal{D}_{tr}^{prv}$ in the clear because the computation is over encrypted (secret-shared) data.

Setup: Simply running a semi-honest 2PC protocol directly between $\mathcal{P}rv_{mal}$ and $\mathcal{V}er$ is not sufficient, because $\mathcal{P}rv_{mal}$ can easily change the outcome of `DistCheck` for p_{req} by flipping its output share. To account for $\mathcal{P}rv_{mal}$, one could use malicious 2PC protocols directly between $\mathcal{P}rv$ and $\mathcal{V}er$, but this is prohibitively expensive. Instead, $\mathcal{P}rv$ and $\mathcal{V}er$ rely on secure outsourced computation to independent and non-colluding servers $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ as done in prior work [516] and in practical deployments [517]. $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ can be instantiated by different companies, which according to data protection laws must protect user data, and thus cannot share their data with each other.

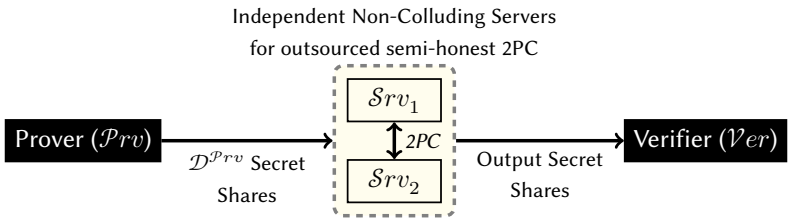


Figure 7.2: **Cryptographic attestation**: $\mathcal{P}rv$ sends the secret shares of $\mathcal{D}^{\mathcal{P}rv}$ to servers $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$. The servers securely compute `DistCheck` for $\mathcal{D}^{\mathcal{P}rv}$ and train $\mathcal{M}_{2pc}$ on it with their secret shares using 2PC. The output shares are sent to $\mathcal{V}er$ for reconstruction.

Method: I consider secret sharing over a ring with Q elements, where a secret input x is split into two shares, x_1 and x_2 such that $x = x_1 + x_2 \mod Q$ [476]. Each of the shares x_1 and x_2 look random, i.e., given only one share, one cannot learn any information about the secret. For (a), `DistCheck` computes the distributional property directly over $\mathcal{D}_{tr}^{prv}$ and compares it with p_{req}. Here, $\mathcal{P}rv$ sends secret shares of $\mathcal{D}_{tr}^{prv}$ to $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$, who jointly perform `DistCheck` by running secure accumulation and comparison using 2PC. For (b), $\mathcal{P}rv$ sends secret shares of the initial model weights to $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ in order to obtain $\mathcal{M}$. Together with the previously obtained shares of $\mathcal{D}_{tr}^{prv}$, $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ jointly run secure training. $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ send the resulting secret shares of `DistCheck` and the final model parameters of the trained model $\mathcal{M}_{2pc}$, to $\mathcal{V}er$. Then, $\mathcal{V}er$ adds the received shares to get the results of `DistCheck` and $\mathcal{M}_{2pc}$'s weights. The *correctness* property of 2PC convinces $\mathcal{V}er$ that both `DistCheck` and the training are executed correctly on $\mathcal{D}_{tr}^{prv}$. I implement secure computation between $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ using CrypTen [476], a framework for efficient privacy-preserving ML that supports semi-honest two-party computation (see Appendix B.4 for security and correctness of the protocols).

Mechanism 3: Hybrid

I later show that the cryptographic mechanism is costly (Table 7.4), while the ML-based mechanism is ineffective, exhibiting unacceptably high false acceptance (FAR) and false rejection (FRR) rates (Table 7.2). Relying solely on either is inadequate. Therefore, I propose a hybrid mechanism that first uses ML-based mechanism with cryptographic mechanism as a fallback. Depending on the application, $\mathcal{V}er$ can fix an acceptably low FAR or FRR:

- **Fixed FAR:** For accepted $\mathcal{P}rv$s, no further action is needed. If the ML-based mechanism fails (FR), $\mathcal{P}rv$s can request re-evaluation with cryptographic mechanism. $\mathcal{V}er$ has no recourse for the FAs (which are low).

- **Fixed FRR:** If ML-based mechanism is rejected, there is no re-appeal since FRR is low (i.e., no recourse for rejected provers). For accepted $\mathcal{P}rv$s, $\mathcal{V}er$ may do a random "spot-check" using cryptographic mechanism to reduce the number of FAs.

Assumptions: I assume FAR and FRR are fixed at 5%. Additionally, $\mathcal{V}er$ uses $\mathcal{M}$ for ML-based mechanism and $\mathcal{M}_{2pc}$ is obtained by 2PC training. I assume that $\mathcal{P}rv$ shares the hyperparameters for training $\mathcal{M}$, such that $\mathcal{M}_{2pc}$ is perfectly equivalent. This can be done by a fidelity check, i.e., sending arbitrary inputs and ensuring outputs from $\mathcal{M}$ and $\mathcal{M}_{2pc}$ are equal.

Method: While the hybrid mechanism is straight-forward for fixed FAR, I present the approach for random spot-checks of accepted $\mathcal{P}rv$s for fixed FRR. Let q be the total FA on $\mathcal{D}_{te}^{\mathcal{V}er}$ from the ML-based mechanism and $\mathcal{N}_{acc}$ denote the number of accepted $\mathcal{P}rv$s. Knowing q, $\mathcal{V}er$ can randomly sample $\mathcal{N}_{spchk}$ spot-checks where $q \leq \mathcal{N}_{spchk} \leq \mathcal{N}_{acc}$. $\mathcal{V}er$ then uses cryptographic mechanism to eliminate any FA in the sampled set thus reducing the overall FAR. To compute the new FAR, I first compute the probability of finding t FAs from the sample of $\mathcal{N}_{spchk}$ provers. I model the probability distribution over FA as hypergeometric distribution which computes the likelihood of selecting t FAs in a sample of $\mathcal{N}_{spchk}$ from a population of q falsely accepted $\mathcal{P}rv$s without replacement:

$$\mathbb{P}(T = t) = \binom{q}{t}\binom{\mathcal{N}_{acc} - q}{\mathcal{N}_{spchk} - t} / \binom{\mathcal{N}_{acc}}{\mathcal{N}_{spchk}}$$

where $t \in [0, q]$. I compute the effective #FAs as

$$\#FA_{new} = \#FA_{old} - t'$$

where $t' = argmax_{t \in [0,q]} \mathbb{P}(T = t)$ and $\mathcal{N}_{spchk}$ will determine the FAR and cost incurred.

7.3 Experimental Setup

I describe the different datasets used and corresponding model architectures, followed by the metrics used for evaluation.

Datasets

I use the datasets, properties, and model architectures following prior work on distribution inference attacks [12].

BONEAGE is an image dataset of hand X-rays, where the task is to predict the patient's age in months. I convert it into a binary classification task based on age. I focus on the proportion of females as the property of interest and consider the following permissible ratios ($\overline{\mathbf{p}}$): [0.2–0.8]. Here, the sensitive attribute is implicit in the metadata.

ARXIV is a directed graph dataset representing citations among computer science ArXiv papers. The task is to predict each paper's subject area. The property used for attestation is the graph's mean node degree. I consider the following permissible range ($\overline{\mathbf{p}}$): [9–17]. The dataset is sampled to achieve a specific mean node degree, which is implicitly encoded.

CENSUS is a tabular dataset with categorical and numerical attributes such as age, race, and education level. The task is to predict whether an individual's annual income exceeds 50K. I use two variants based on the property of interest: (a) CENSUS-R, which considers the proportion of white individuals, and (b) CENSUS-S, which considers the proportion of females. For both, I use the permissible range ($\overline{\mathbf{p}}$): [0.0–1.0]. In both CENSUS-S and CENSUS-R, the sensitive attributes are explicitly included.

Table 7.1: **Summary of datasets:** I describe the classification task, the attested property, whether it is explicit in $\mathcal{D}_{tr}$, and the dataset sizes for $\mathcal{V}er$ and $\mathcal{P}rv$ before sampling.

	ARXIV	**BONEAGE**	**CENSUS-R**	**CENSUS-S**
Type	Graph	Images	Tabular	Tabular
Task	Node	Age	Income	Income
Property	Node degree	Ratio: females	Ratio: Whites	Ratio: females
$\mathcal{V}er$ **Size**	40,000	1800	1800	3200
$\mathcal{P}rv$ **Size**	97,000	3600	1800	3200
Explicit?	No	No	Yes	Yes

Model Architectures

BONEAGE model is a pre-trained DenseNet model [518] followed by a three-layer network with sizes [128, 64, 1] and ReLU activations. I train the model for 100 epochs with a batch size of 8192, a learning rate of 0.001, and a weight decay of 1e–4.

ARXIV model is a four-layer graph convolutional network that maps input graph data to low-dimensional embeddings for node classification. The layer sizes are [256, 256, 256, 40] with ReLU activations, and I apply dropout with a probability of 0.5. The model is trained for 100 epochs with a learning rate of 0.01 and a weight decay of 5e–4.

CENSUS model is a three-layer neural network with hidden dimensions [32, 16, 8] and ReLU activations. I train it for 100 epochs with a learning rate of 0.01, a batch size of 64, and no weight decay.

f_{atst} **model** is a permutation-invariant network based on the DeepSets architecture [519], as used in distribution inference literature [12, 122]. It produces representations of input model parameters that are invariant to neuron ordering within layers. Prior work by Suri and Evans [12] suggests that the first layer captures distributional properties more effectively than later layers; accordingly, I use the first layer's parameters as input to f_{atst}, which outputs the property of the training data.

Metrics

I describe different metrics to measure the effectiveness of inference-based attestation. A model trained on a dataset with p_{req} is considered as the positive class.

- **Accuracy** indicates the utility of $\mathcal{P}rv$'s model and $\mathcal{V}er$'s shadow models on the primary classification task.
- **True Acceptance Rate (TAR)** measures the fraction of models where $\mathcal{V}er$ correctly attests that $\mathcal{M}$ was indeed trained from a dataset with p_{req}.
- **True Rejection Rate (TRR)** measures the fraction of models where $\mathcal{V}er$ correctly rejects attestation of $\mathcal{M}$ w.r.t. p_{req}.
- **False acceptance rate (FAR) and false rejection rate (FRR)** quantify how often $\mathcal{V}er$ incorrectly accepts or rejects attestations, respectively.
- **Equal Error Rate (EER)** indicates the value at which the FAR and FRR are equal.

TAR and TRR should ideally be 1.00 while FAR, FRR, and EER should be 0.00.

- **Communication Cost:** For the cryptographic mechanism, I indicate computation cost (ω_{crpt}^{comp}) as the execution time for `DistCheck` and secure model training; and communication cost (ω_{crpt}^{comm}) as the amount of data transferred during attestation. For the hybrid attestation mechanism with fixed FAR where $\mathcal{P}rv$ relies on cryptographic fallback, the expected cost is $\mathbb{P}_{inf} \times \omega_{inf} + \mathbb{P}_{crpt} \times \omega_{crpt}$ where $\mathbb{P}_{inf} = 1$. As $\omega_{inf} \ll \omega_{crpt}$, the cost reduces to $\mathbb{P}_{crpt} \times \omega_{crpt}$. Similarly, for fixed FRR, $\mathcal{V}er$ conducts spot-checks with a probability of $\mathbb{P}_{spchk}$. The expected cost in this case is $\mathbb{P}_{spchk} \times \omega_{crpt}$. Both $\mathbb{P}_{crpt}$ and $\mathbb{P}_{spchk}$ are computed on $\mathcal{D}_{te}^{\mathcal{V}er}$.

7.4 Experimental Evaluation

I evaluate ML-based and cryptographic mechanisms and identify their limitations, then evaluate the hybrid mechanism. I focus on effectiveness (**R1**), efficiency (**R2**), and robustness (**R5**), and discuss the remaining requirements towards the end of Section 7.4.

7.4.1 Mechanism 1: ML-based

Effectiveness (R1): I first evaluate the effectiveness of f_{atst} in distinguishing between models trained on $\mathcal{D}_{tr}^{prv}$ with p_{req} and $!p_{req}$ using AUC score under FAR-TAR curves (see Appendix B.1). I find that for some p_{req} values, f_{atst} is less effective. Hence, I relax the attestation requirement to exactly match p_{req} by increasing the window size to ± 1 (i.e., classify between {p_{req}-1, p_{req}, p_{req}+1} and $!${p_{req}-1, p_{req}, p_{req}+1})[6]. Based on the results, I identify the best window sizes on $\mathcal{D}_{te}^{\mathcal{V}er}$: ± 1 for all p_{req} for BONEAGE and ARXIV; 0 for the edge p_{req} values (i.e., 0.00 and 1.00) and ± 1 for all middle p_{req} values for CENSUS-R and CENSUS-S. $\mathcal{V}er$ can make these decisions on $\mathcal{D}_{te}^{\mathcal{V}er}$ before finalizing f_{atst}.

Assuming that $\mathcal{V}er$ fixes (a) FAR, or (b) FRR at 5%, I present the corresponding TRR and TAR in Table 7.2 on $\mathcal{D}_{\text{P}}^{vrf}$. At either end of the spectrum of p_{req} values, attestation is effective (high TRR/TAR). However, I observe a high FAR and FRR for the middle p_{req} values indicating that attestation is less effective. Furthermore, I provide EER values that, for specific p_{req} values, demonstrate lower rates than both FAR and FRR. This implies the existence of a more optimal threshold than the currently used 5%. In summary, ML-based mechanism is *ineffective for certain p_{req} values* and cannot be used on its own.

Efficiency (R2): $\mathcal{V}er$ trains multiple shadow models and f_{atst}. I measure the total training time for 10 attestation classifiers and 1000 shadow models on a single NVIDIA A100 GPU. Training f_{atst} took a total of 200 mins for BONEAGE; 12 mins for ARXIV, 6 mins for CENSUS-S and CENSUS-R. Training 1000 shadow models took a total of 173 mins for BONEAGE, 123 mins for ARXIV and 50 mins for CENSUS-S and CENSUS-R.

BONEAGE, being an image dataset trained on a large neural network, takes the maximum time for both f_{atst} and shadow models. On the other hand, CENSUS has a small number of tabular data records and with a small MLP classifier, takes the least training time. Note that this is a one-time cost which can be parallelized among multiple GPUs. Hence, $\mathcal{V}er$'s cost for ML-based mechanism is reasonable. f_{atst} can then be used for multiple attestations for the same property and has to be trained on a new property only if p_{req} changes.

[6] I continue to use p_{req} and $!p_{req}$ to refer to these windows.

Table 7.2: **TAR and TRR with 5% thresholds for FAR and FRR, respectively, along with EER across different p_{req} windows on $\mathcal{D}^{vrf}_{\mathbf{P}}$**: The p_{req} value within the window is indicated in **bold**. Edge p_{req} values have higher effectiveness than middle p_{req} values due to higher distinguishability in first layer parameters [12].

ARXIV

p_{req} Range	TAR	TRR	EER
{**9**, 10}	1.00	0.99	0.02
{9, **10**, 11}	1.00	1.00	0.01
{10, **11**, 12}	0.24	0.83	0.16
{11, **12**, 13}	0.61	0.68	0.19
{12, **13**, 14}	0.78	0.85	0.10
{13, **14**, 15}	0.92	0.93	0.07
{14, **15**, 16}	0.87	0.90	0.08
{15, **16**, 17}	1.00	1.00	0.00
{16, **17**}	1.00	1.00	0.00

BONEAGE

p_{req} Range	TAR	TRR	EER
{**0.20**, 0.30}	0.96	0.96	0.03
{0.20, **0.30**, 0.40}	0.99	1.00	0.02
{0.30, **0.40**, 0.50}	0.87	0.88	0.09
{0.40, **0.50**, 0.60}	0.53	0.65	0.21
{0.50, **0.60**, 0.70}	0.39	0.72	0.25
{0.60, **0.70**, 0.80}	0.98	0.98	0.03
{0.70, **0.80**}	0.95	0.95	0.05

CENSUS-S

p_{req} Range	TAR	TRR	EER
{**0.00**}	1.00	1.00	0.00
{0.00, **0.10**, 0.20}	0.49	0.49	0.19
{0.10, **0.20**, 0.30}	0.70	0.72	0.14
{0.20, **0.30**, 0.40}	0.23	0.56	0.25
{0.30, **0.40**, 0.50}	0.12	0.30	0.37
{0.40, **0.50**, 0.60}	0.13	0.23	0.41
{0.50, **0.60**, 0.70}	0.15	0.22	0.34
{0.60, **0.70**, 0.80}	0.12	0.26	0.35
{0.70, **0.80**, 0.90}	0.59	0.58	0.19
{0.80, **0.90**, 1.00}	0.60	0.59	0.19
{**1.00**}	1.00	1.00	0.00

CENSUS-R

p_{req} Range	TAR	TRR	EER
{**0.00**}	1.00	1.00	0.00
{0.00, **0.10**, 0.20}	0.21	0.64	0.19
{0.10, **0.20**, 0.30}	0.75	0.89	0.10
{0.20, **0.30**, 0.40}	0.22	0.59	0.23
{0.30, **0.40**, 0.50}	0.14	0.16	0.39
{0.40, **0.50**, 0.60}	0.10	0.15	0.42
{0.50, **0.60**, 0.70}	0.13	0.26	0.39
{0.60, **0.70**, 0.80}	0.05	0.36	0.32
{0.70, **0.80**, 0.90}	0.65	0.77	0.13
{0.80, **0.90**, 1.00}	0.35	0.41	0.26
{**1.00**}	1.00	1.00	0.00

Robustness (R5): $\mathcal{P}rv_{mal}$ can fool $\mathcal{V}er$ by modifying $\mathcal{M}$'s first layer parameters ($\mathcal{M}^1_p$) to trigger FA. $\mathcal{P}rv_{mal}$ adds adversarial noise δ to the first layer parameters: $\mathcal{M}^1_p + \delta$, where $\delta = argmax_{||\delta||_p<\epsilon_{rob}} l(\, f_{atst}(\mathcal{M}^1_p+\delta\,), p_{req}\,)$, l is f_{atst}'s loss and $||\cdot||_p$ is the l_p norm to ensure that δ minimizes accuracy degradation. Since $\mathcal{P}rv_{mal}$ does not have access to f_{atst}, they train a "substitute model" on $\mathcal{D}^{prv}_{tr}$ that mimics f_{atst}. For worst-case analysis under the attack, I assume that the substitute model's architecture is the same as f_{atst}. δ is then computed with respect to this substitute model and the FA are expected to transfer to f_{atst}. To restore any $\mathcal{M}$'s accuracy loss, $\mathcal{P}rv_{mal}$ can freeze $\mathcal{M}^1_p$ and fine-tune the remaining layers. I empirically evaluate this and confirm that accuracy of model after fine-tuning is close to the original accuracy while still being able to fool the attestation (results in Appendix B.3). I use Autoattack [387] with

$\epsilon_{rob} = 8/255$, and L_∞ norm for the distance function. As $\mathcal{P}rv_{mal}$ has access to the models shared with $\mathcal{V}er$, I evaluate on $\mathcal{D}_{\mathsf{P}}^{vrf}$.

Table 7.3: **Robustness against first layer parameter perturbations with and without a defense:** Utility (ϕ_{acc}) is calculated using AUC on FAR-TAR for a clean dataset. FAR $\rightarrow$ lack of robustness. Green $\rightarrow \phi_{\text{acc}}$ decreases or FAR $< 5\%$, red $\rightarrow \phi_{\text{acc}}$ increases or FAR $\geq 5\%$.

ARXIV

p_{req} **Range**	w/o Defence		w/ Defence	
	ϕ_{acc}	FAR	ϕ_{acc}	FAR
{**9**, 10}	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	0.00 ± 0.00
{9, **10**, 11}	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
{10, **11**, 12}	0.92 ± 0.00	1.00 ± 0.00	0.88 ± 0.00	0.00 ± 0.00
{11, **12**, 13}	0.96 ± 0.00	1.00 ± 0.00	0.96 ± 0.00	0.00 ± 0.00
{12, **13**, 14}	0.93 ± 0.00	1.00 ± 0.00	0.96 ± 0.01	0.00 ± 0.00
{13, **14**, 15}	0.99 ± 0.00	1.00 ± 0.00	0.96 ± 0.01	0.00 ± 0.00
{14, **15**, 16}	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	0.00 ± 0.00
{15, **16**, 17}	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
{16, **17**}	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	0.00 ± 0.00

BONEAGE

p_{req} **Range**	w/o Defence		w/ Defence	
	ϕ_{acc}	FAR	ϕ_{acc}	FAR
{**0.20**, 0.30}	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.00 ± 0.00
{0.20, **0.30**, 0.40}	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.00 ± 0.00
{0.30, **0.40**, 0.50}	0.92 ± 0.00	1.00 ± 0.00	0.92 ± 0.00	0.00 ± 0.00
{0.40, **0.50**, 0.60}	0.86 ± 0.00	1.00 ± 0.00	0.84 ± 0.00	0.00 ± 0.01
{0.50, **0.60**, 0.70}	0.87 ± 0.00	0.28 ± 0.00	0.85 ± 0.00	0.25 ± 0.00
{0.60, **0.70**, 0.80}	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.02 ± 0.00
{0.70, **0.80**}	0.95 ± 0.00	0.04 ± 0.00	0.95 ± 0.00	0.00 ± 0.00

CENSUS-S

p_{req} **Range**	w/o Defence		w/ Defence	
	ϕ_{acc}	FAR	ϕ_{acc}	FAR
{**0.00**}	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
{0.00, **0.10**, 0.20}	0.83 ± 0.01	0.26 ± 0.05	0.82 ± 0.01	0.01 ± 0.01
{0.10, **0.20**, 0.30}	0.92 ± 0.00	0.12 ± 0.02	0.92 ± 0.01	0.04 ± 0.01
{0.20, **0.30**, 0.40}	0.78 ± 0.01	0.11 ± 0.02	0.79 ± 0.00	0.10 ± 0.02
{0.30, **0.40**, 0.50}	0.66 ± 0.00	0.34 ± 0.10	0.66 ± 0.01	0.29 ± 0.03
{0.40, **0.50**, 0.60}	0.67 ± 0.01	0.39 ± 0.02	0.67 ± 0.00	0.22 ± 0.03
{0.50, **0.60**, 0.70}	0.62 ± 0.01	0.19 ± 0.02	0.62 ± 0.01	0.14 ± 0.01
{0.60, **0.70**, 0.80}	0.68 ± 0.00	0.48 ± 0.01	0.68 ± 0.01	0.35 ± 0.04
{0.70, **0.80**, 0.90}	0.89 ± 0.00	0.32 ± 0.02	0.89 ± 0.00	0.32 ± 0.05
{0.80, **0.90**, 1.00}	0.89 ± 0.01	0.65 ± 0.03	0.89 ± 0.00	0.22 ± 0.03
{**1.00**}	1.00 ± 0.00	0.02 ± 0.00	1.00 ± 0.00	0.00 ± 0.00

CENSUS-R

p_{req} **Range**	w/o Defence		w/ Defence	
	ϕ_{acc}	FAR	ϕ_{acc}	FAR
{**0.00**}	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
{0.00, **0.10**, 0.20}	0.75 ± 0.01	0.82 ± 0.02	0.77 ± 0.01	0.35 ± 0.03
{0.10, **0.20**, 0.30}	0.95 ± 0.00	0.60 ± 0.03	0.95 ± 0.00	0.08 ± 0.01
{0.20, **0.30**, 0.40}	0.71 ± 0.01	0.83 ± 0.02	0.71 ± 0.01	0.13 ± 0.05
{0.30, **0.40**, 0.50}	0.75 ± 0.00	0.82 ± 0.07	0.74 ± 0.00	0.05 ± 0.01
{0.40, **0.50**, 0.60}	0.64 ± 0.00	0.79 ± 0.05	0.63 ± 0.00	0.11 ± 0.03
{0.50, **0.60**, 0.70}	0.60 ± 0.01	0.31 ± 0.02	0.60 ± 0.00	0.31 ± 0.02
{0.60, **0.70**, 0.80}	0.83 ± 0.00	0.30 ± 0.02	0.82 ± 0.01	0.26 ± 0.01
{0.70, **0.80**, 0.90}	0.96 ± 0.00	0.10 ± 0.01	0.96 ± 0.01	0.19 ± 0.00
{0.80, **0.90**, 1.00}	0.70 ± 0.01	0.46 ± 0.03	0.75 ± 0.01	0.34 ± 0.02
{**1.00**}	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00

Attack Success: $\mathcal{P}rv_{mal}$ wins if f_{atst} incorrectly classifies perturbed models as having been trained with p_{req}. I measure the attack success using FAR. Note that the FAR here is restricted to $\mathcal{D}_{\mathsf{P}}^{vrf}$ containing models with adversarial noise. Under "w/o Defence" in Table 7.3, the high FAR values indicate that the attack is indeed successful (f_{atst} is not robust).

Improving Robustness: I propose adversarial training of f_{atst} where $\mathcal{V}er$ includes models with adversarial noise to train f_{atst}. The goal is to reduce FAR on perturbed models while retaining utility on clean $\mathcal{D}_{\mathsf{P}}^{vrf}$. I present the results of adversarial training under "w/ Defence" in Table 7.3. First, the FAR values in "w/ Defence" are lower (than in "w/o Defence"). Hence, adversarial training of f_{atst} successfully mitigates the perturbation attack, thus making ML-based

mechanism adversarially robust, satisfying robustness (R5). Second, the difference in utility on clean $\mathcal{D}_{\mathsf{P}}^{vrf}$ (measured using AUC score under FAR-TAR curves indicated by ϕ_{acc}) between "w/ Defence" and "w/o Defence" is small. I also evaluate the effectiveness using TRR, TAR, and EER on clean $\mathcal{D}_{\mathsf{P}}^{vrf}$ (see Appendix B.2). I use robust f_{atst} in the remainder of this chapter. Similar to f_{atst}, robust f_{atst} is still ineffective for some p_{req}.

Takeaway: ML-based mechanism is *efficient* (R2) and *robust* (R5) but *not effective*.

7.4.2 Mechanism 2: Cryptographic

Effectiveness (R1): The cryptographic mechanism operates over $\mathcal{D}_{tr}^{prv}$ confidentially, to correctly check if the distributional properties match p_{req}. Hence, I have *zero FAR and FRR*, In other words, cryptographic mechanism is perfectly effective.

Efficiency (R2): I use protocols for semi-honest parties. I present the computation and communication cost for a single cryptographic attestation in Table 7.4. I indicate the costs for BONEAGE and CENSUS but omit the evaluation on ARXIV as there are no PyTorch frameworks for secure GNN training, which is required by the CrypTen library. I observe that the cost for `DistCheck` is low, but the cost for secure ML training is high. Hence, the cryptographic mechanism is difficult to use in practice for multiple $\mathcal{P}rv$s.

Table 7.4: **Computation (ω_{crpt}^{comp}) and communication costs (ω_{crpt}^{comm}) of cryptographic attestation** for a single $\mathcal{P}rv$ averaged over 20 runs.

Datasets	ω_{crpt}^{comp} (s)		ω_{crpt}^{comm} (GB)	
	DistCheck	Training	DistCheck	Training
BONEAGE	1.30 ± 0.05	1367.31 ± 27.95	0.01	228.54
CENSUS-R	1.54 ± 0.15	1081.00 ± 17.00	0.01	874.06
CENSUS-S	1.68 ± 0.15	2109.78 ± 65.20	0.01	1438.38

Robustness (R5): Using outsourcing, $\mathcal{P}rv$'s inputs are secret-shared between $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ who learn nothing (non-colluding assumption) and have no incentive to cheat (semi-honest assumption). Also, $\mathcal{P}rv$ *only* performs the input sharing of $\mathcal{D}_{tr}^{prv}$ and initial model weights, thus, $\mathcal{P}rv$ cannot cheat during proof generation. Hence, this attestation is robust.

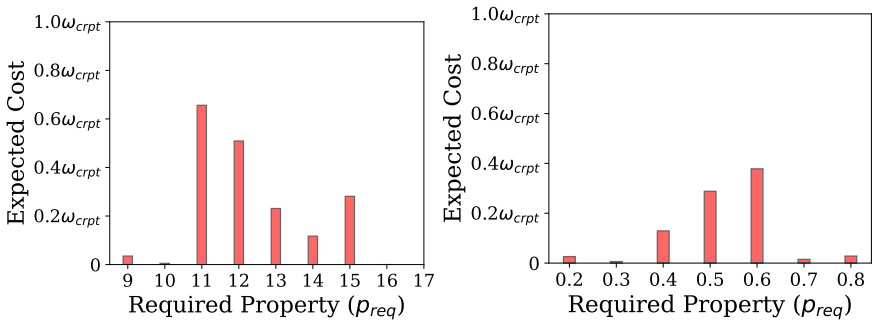
Takeaway: Cryptographic attestation using MPC is effective (R1) and robust (R5), but it is *not efficient* which makes it challenging to scale to multiple verifiers.

7.4.3 Mechanism 3: Hybrid

I present the effectiveness of the hybrid attestation mechanism with a fixed FAR (or FRR), its impact on the respective FRR (or FAR) and the expected cost.

Fixed FAR Analysis

Recall that when FAR is fixed, rejected $\mathcal{P}rv$s can request re-evaluation using the cryptographic mechanism as a fallback.



(a) ARXIV (b) BONEAGE

(c) CENSUS-R (d) CENSUS-S

Figure 7.3: **Fixed FAR@5%: Expected cost on $\mathcal{D}_{\mathbf{P}}^{vrf}$:** ω_{crpt} is a placeholder for ω_{crpt}^{comp} and ω_{crpt}^{comm}. Expected cost is less than cryptographic attestation (=ω_{crpt}).

Effectiveness (R1): The hybrid mechanism will *not change the 5% fixed FAR value*. I compute the effective FRR on using cryptographic mechanism as a fallback. In practice, only $\mathcal{P}rv$s with FR have the incentive to request re-evaluation using cryptographic mechanism. If such $\mathcal{P}rv$s undergo re-evaluation, *FRR is zero* as it will rectify erroneous decisions.

Efficiency (Expected Cost) (R2): I evaluate the expected cost on $\mathcal{D}_{\mathsf{P}}^{vrf}$. This gives the actual estimate of the cost incurred during attestation. I assume that $\mathcal{P}rv$s are rational, so only $\mathcal{P}rv$s with FR will request a re-evaluation using cryptographic mechanism. Here, to compute the expected cost, $\mathbb{P}_{crpt} = \frac{\mathcal{N}_{rej}}{\mathcal{N}}$, where $\mathcal{N}_{rej}$ is the total number of rejected $\mathcal{P}rv$s.

I present the expected cost in Figure 7.3, where the values of ω_{crpt}—a placeholder for ω_{crpt}^{comp} or ω_{crpt}^{comm}—are taken from Table 7.4. Compared to the cryptographic mechanism, which has an expected cost of ω_{crpt}, the hybrid mechanism has a lower cost across different datasets and p_{req} values. Additionally, since $\mathbb{P}_{crpt}$ depends on $\mathcal{N}_{rej}$ computed by the ML-based mechanism, extreme p_{req} values (where the ML-based mechanism is more effective) result in lower expected costs than intermediate p_{req} values.

Fixing FRR

Here, recall that $\mathcal{V}er$ conducts random spot-checks to reduce FAR.

Effectiveness (R1): $\mathcal{V}er$'s choice of $\mathcal{N}_{spchk}$ determines the FAR. No spot-checks results in the same FAR as the ML-based mechanism, while spot-checking all accepted provers gives zero FAR.

Efficiency (R2): The expected cost incurred per $\mathcal{P}rv$ increases with $\mathcal{N}_{spchk}$. No spot-checks corresponds to no expected cryptographic cost while spot-checks for all accepted $\mathcal{P}rv$s incurs a high expected cost. Hence, $\mathcal{V}er$ decides $\mathcal{N}_{spchk}$ based on their application's requirement.

I present the trade-off between FAR and expected cost of the cryptographic mechanism by varying $\mathcal{N}_{spchk}$ on $\mathcal{D}_{te}^{\mathcal{V}er}$, as shown in Figure 7.4. I use $\mathbb{P}_{spchk} = \frac{\mathcal{N}_{spchk}}{\mathcal{N}}$ to compute the expected cost. Increasing $\mathcal{N}_{spchk}$ increases the expected cost while decreasing FAR. Using Figure 7.4, $\mathcal{V}er$ can select an appropriate value for $\mathcal{N}_{spchk}$. Once $\mathcal{V}er$ chooses a suitable $\mathcal{N}_{spchk}$ using $\mathcal{D}_{te}^{\mathcal{V}er}$, the corresponding cost and FAR can be read from a plot (similar to Figure 7.4) for $\mathcal{D}_{\mathsf{P}}^{vrf}$ (see Appendix B.5).

Carefully choosing $\mathcal{N}_{spchk}$ leads to a notable reduction in FAR compared with $\mathcal{N}_{spchk}$= 0 (x-axis). Additionally, the expected cost is lower compared to conducting spot-checks for all accepted $\mathcal{P}rv$s and the purely cryptographic mechanism (y-axis where $\mathcal{N}_{spchk}$= 0).

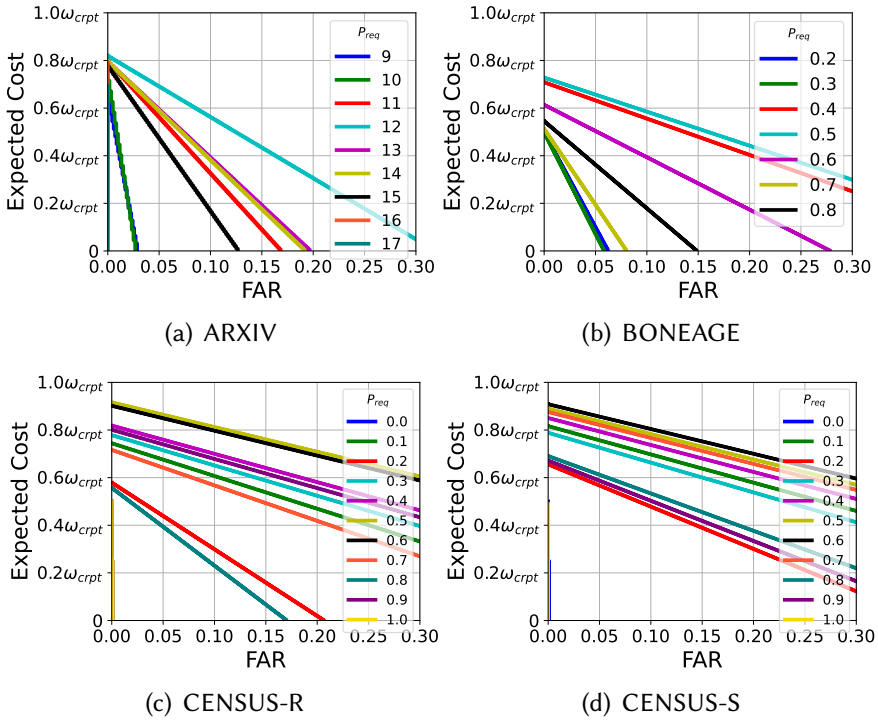


Figure 7.4: **Fixed FRR@5%: Trade-off between FAR and expected cost on varying $\mathcal{N}_{spchk}$ on $\mathcal{D}_{te}^{\mathcal{V}er}$:** Expected cost is less than cryptographic mechanism (=ω_{crpt}) and effectiveness is better than ML-based mechanism. ω_{crpt} is a placeholder for both ω_{crpt}^{comp} and ω_{crpt}^{comm}.

Takeaway: Hybrid attestation is more effective than ML-based approaches and cheaper than purely cryptographic methods, but requires joint training with each verifier, limiting scalability to multiple verifiers. Thus, this is still not practical.

Summary of Software-based ML Property Attestations

Based on the above study, I summarize the limitations of various software-based mechanisms with respect to different requirements (R1–R5). I use ○ to indicate "not satisfied," ⊛ for "partially satisfied," and ● for "fully satisfied" (summarized in Table 7.5).

- ***ML-based Mechanisms***: I show that re-purposing inference attacks as attestation mechanisms falls short of meeting all requirements. Although this approach is efficient, with low query cost (R2 → ●), and scalable to multiple verifiers (R3 → ●), it is limited to specific

properties and cannot, on its own, support others such as proof of evaluation and inference (R4 → ○). Furthermore, these mechanisms are not effective (R1 → ○) and exhibit poor robustness, requiring additional optimization (R5 → ⊛). Prior work has also shown that both membership inference and distribution inference are unreliable for auditing [520, 521, 4].

- ***Cryptographic Mechanisms***: I show that **MPC** requires online interaction between $\mathcal{P}rv$ and $\mathcal{V}er$, which limits scalability and efficiency, as it requires retraining the model for each verification [4] (R2, R3 → ○). Primitives such as **ZKPs** are *effective* (R1 → ●), *scalable* to multiple verifiers (R3 → ●), and *robust* (R5 → ●). However, they are computationally expensive and do not efficiently scale to neural networks (R2 → ○). For instance, proof of training using ZKPs for a simple logistic regression model on a dataset of 2^{18} records with 1024 attributes and a batch size of 2014 takes 72 seconds per epoch, including both training and proof generation [503]. Finally, ZKPs lack versatility, as they only support properties that can be expressed within the ZKP framework [503], limiting their applicability to complex functions such as nonlinear activations (R4 → ○).
- ***Hybrid Mechanism***: The hybrid mechanism is *more effective than ML-based approaches* but *less effective than cryptographic ones* (R1 → ⊛). It is also *more efficient than cryptographic mechanisms*, though *less efficient than ML-based solutions* (R2 → ⊛). However, its reliance on ML introduces a residual risk of evasion by malicious provers (R5 → ⊛). Moreover, the hybrid approach is *not scalable*, as it requires joint training with verifiers for each attestation (R3 → ○), and it *lacks versatility*, similar to the other two mechanisms (R4 → ○).

Table 7.5: **Summary of requirements satisfied by various software-based ML property attestation mechanisms**: ○ → Not satisfied; ⊛ → Partially satisfied; ● → fully satisfied.

Technique	R1 Effective	R2 Efficient	R3 Scalable	R4 Versatile	R5 Robust
ML-based	○	●	●	○	⊛
MPC	●	○	○	○	●
ZKP	●	○	●	○	●
Hybrid	⊛	⊛	○	○	⊛

Takeaway: None of the software-based ML property attestation mechanisms meet all requirements (negative response for RQ2.1). Hence, we need to explore other mechanisms.

7.5 Discussions

Privacy Attacks by Verifier for ML-based Mechanism: Given access to $\mathcal{M}$, $\mathcal{V}er$ may perform various privacy attacks, as it is assumed to be "honest-but-curious," i.e., it may use any available information but does not deviate from the specified protocol. To mitigate this, $\mathcal{P}rv$ can apply differential privacy `DiffPriv` to reduce privacy risks while preserving attestation capability. Prior work has shown that distribution inference attacks can achieve higher accuracy under `DiffPriv`, assuming $\mathcal{V}er$ knows `DiffPriv` hyperparameters, which is a reasonable assumption in the attestation setting [121]. Therefore, I expect ML-based attestation to be *more effective* when `DiffPriv` is applied.

Malicious Verifier for Cryptographic Mechanism: A malicious $\mathcal{V}er$ does not interact with the protocol directly but only receives output shares from the outsourcing servers. Hence, it cannot manipulate the protocol to trigger a false rejection. In contrast, for ML-based attestation, a malicious $\mathcal{V}er$ could potentially use an attestation classifier for a different property. In the context of regulatory compliance, it is reasonable to assume that $\mathcal{V}er$ is honest-but-curious and does not deviate from the specified protocol. A malicious $\mathcal{V}er$ can be accommodated in several ways: (i) relying solely on a cryptographic mechanism (which is expensive), (ii) relying on a hardware-based trusted execution environment (TEEs), or (iii) training $\mathcal{M}$ with `DiffPriv` (which reduces model utility).

Contribution in Support of the Thesis Statement

Practitioners cannot make verifiable claims about ML properties using ML property cards. I propose ML property attestation mechanisms that allow provers (e.g., model trainers) to demonstrate ML properties to verifiers (e.g., regulators). These attestations can help make ML property cards verifiable. I show that current software-based mechanisms fall short: ML-based mechanisms lack effectiveness and robustness, while cryptographic mechanisms lack efficiency, versatility, and scalability (a negative answer to **RQ2.1**). Thus, alternative mechanisms are needed to support the thesis statement.

Chapter 8

Hardware-based ML Property Attestation Mechanisms

Overview

Given that software-based ML property attestation mechanisms do not meet all requirements (**R1**–**R5** in Chapter 7), I now explore whether hardware-based mechanisms using trusted execution environments (TEEs) can be a practical solution. Recent advances in confidential computing have made it possible to run and train ML models inside TEEs, which provide an efficient remote attestation mechanism. I propose `Laminator`, the first framework to use TEEs to provide verifiable ML property cards via hardware-based ML property attestation. `Laminator` is effective in attesting the required properties, efficient in terms of overhead, scalable to large numbers of verifiers, versatile across properties in the ML pipeline, and robust against malicious provers. Overall, I show that hardware-based ML property attestation mechanisms satisfy all requirements, thereby answering **RQ2.2** and supporting the thesis statement. I discuss how these attestations can be extended to demonstrate that practitioners have evaluated and accounted for the unintended interactions identified in **Part-1**.

As shown in Chapter 7, current software-based property attestation mechanisms have various drawbacks in terms of effectiveness, efficiency, scalability, versatility, and robustness. As an alternative, I consider trusted execution environments (TEEs), which are widely available in commercially deployed *confidential computing* environments [522]. These mechanism already provide hardware-based *remote attestation*, allowing a system or software configuration to be proven to a remote verifier ($\mathcal{V}er$). This capability has been extended to more general property attestation, such as verifying properties of software running inside TEEs [481].

Recent developments have also made it possible to efficiently run and train ML models inside TEEs [523, 524]. TEEs have been used for both training [525] and inference [526] (see the survey by Mo et al. [527]). While these approaches primarily focus on data confidentiality, verifiability of ML properties requires integrity guarantees that have not yet been fully addressed. This raises the following research question: (**RQ2.2**) *can hardware-based ML property attestation mechanisms address the limitations of software-based ones, and meet all requirements?*

I propose `Laminator`, the first framework for *hardware-based ML property attestation* to provide verifiable ML property cards. I use TEEs to establish *bindings* between key properties of a prover's ($\mathcal{P}rv$) training dataset ($\mathcal{D}_{tr}^{prv}$) and a model ($\mathcal{M}$) during training, evaluation, and inference. These bindings are signed by the TEE's secret attestation key, generating ML property attestations. When combined with signed external certificates from trusted certifiers (e.g., about data quality from a trusted data provider like CIFAR), these attestations enable verifiers to draw conclusions about the models, their behavior, or their training datasets. These attestations can make ML property cards verifiable. I claim the following contributions:

1. `Laminator`[1], the *first* framework to generate *verifiable ML property cards* using hardware-based attestations throughout training, evaluation, and inference; (Sections 8.1 and 8.2)
2. showing that `Laminator` meets all requirements using four different types of attestations: *effective*, *efficient*, *scalable*, *versatile*, and *robust*. (Sections 8.3 and 8.4)

8.1 Hardware-based Attestations for Property Cards

I present various ML property attestations, each obtained by executing the relevant ML operations inside a TEE, and then including the attestations in the ML property card. Finally, I discuss how $\mathcal{V}er$ can combine different attestations to draw meaningful conclusions covering the entire ML pipeline (Section 8.1).

Threat Model and Assumptions: Following the threat model used for TEEs, I assume that $\mathcal{P}rv$'s entire computing platform except for the TEE is untrusted, including the privileged code such as the operating system and hypervisor, which are controlled by $\mathcal{P}rv$. I assume two roots of trust, signature verification keys belonging to (a) the manufacturer of the TEE (e.g., Intel) who certifies the keys used by enclaves to sign attestations (or attestation keys), and (b) trusted certifiers (e.g., CIFAR) that provide attestations of computational properties (with trust in the property grounded in the trusted certifier rather than the TEE), and certificates of non-computational properties (e.g., data provenance). As long as $\mathcal{V}er$ trusts them, $\mathcal{V}er$ can trust any attestations or certificates linked to them.

[1]Code: https://github.com/ssg-research/laminator

Format for Hardware-based Attestations: An attestation mechanism (TEE) allows to execute code (for function f) to transform inputs $i_1, i_2, ...$ into outputs $o_1, o_2, ...$ while guaranteeing a certain assertion $Asrt(i_1, i_2, ..., o_1, o_2, ...)$. This guarantee is evidenced by a signature $\text{Sign}_{\text{sk}}(\texttt{'AttX'}, \cdots)$ generated by the TEE only if the assertion holds. Here, $\text{Sign}_{\text{sk}}(\cdot)$ represents a signature created using a private (or attestation) key sk which is not available outside the TEE, and 'AttX' indicates the attestation name (e.g., PoT for proof of training). I describe attestations in the form shown in Attestation 1. These attestations form the basis of Laminator.

Attestation 1: Sample attestation

Input: $i_1, i_2, ...$

Computation: $o_1, o_2, ... \xleftarrow{\$} f(i_1, i_2, ...)$

Output: $o_1, o_2, ..., \texttt{h}(i_1), \texttt{h}(i_2), ... \text{Sign}_{\text{sk}}(\texttt{'AttX'}, \texttt{h}(i_1), \texttt{h}(i_2), ..., \texttt{h}(o_1), \texttt{h}(o_2), ...)$

Assertion: $Asrt(i_1, i_2, ..., o_1, o_2, ...)$

where $\texttt{h}(\cdot)$ is a cryptographic hash function, which are signed using the attestation key sk (i.e., attestation). Not all attestations include all fields. The attestation outputs may include hashes of inputs (e.g., $\texttt{h}(i_1)$, $\texttt{h}(i_2)$,...) that are confidential and unavailable to $\mathcal{V}er$. For simplicity, I assume that all inputs across various attestations are unavailable to $\mathcal{V}er$: $\mathcal{M}$, model architecture $\mathcal{M}_{ar}$, training configuration $\mathcal{T}$, $\mathcal{P}rv$'s test dataset $\mathcal{D}_{te}^{prv}$, and training dataset $\mathcal{D}_{tr}^{prv}$. Hence, the hashes of these inputs are included in the attestation output, $\mathcal{V}er$ can later check against the hashes from public commitments. The same applies to attestation outputs; however, I assume that the outputs are public, and therefore, their hashes are not included in the attestation outputs.

Attestations for Datasheets

I first present an example of dataset attestation required for verifiable datasheets: proof of the attribute distribution of $\mathcal{D}_{tr}^{prv}$. I hereafter refer to a cryptographic hash function as $h(\cdot)$.

Proof of attribute distribution (DistAtt) attests the distributional properties P (such as gender ratio) of $\mathcal{D}_{tr}^{prv}$ by providing an assertion that they have a specific value. This value can later be checked against $\mathcal{V}er$'s requirements (e.g., in Chapter 7). Distributional properties can be defined across: (i) a single attribute (e.g., male-to-female ratio), (ii) multiple attributes (e.g., male-to-female *and* caucasian-to-non-caucasian ratios), and (iii) joint properties (i.e., fraction of data records for a given classification label for different subgroups). For a given training dataset used by $\mathcal{P}rv$ ($\mathcal{D}_{tr}^{prv}$), Laminator measures the properties to be attested and

generates the following as an output: the property P, $\mathtt{h}(\mathcal{D}^{prv}_{tr})$, and the attestation which is $\mathrm{Sign}_{\mathrm{sk}}(\texttt{'DistAtt'}, \mathtt{P}, \mathtt{h}(\mathcal{D}^{prv}_{tr}))$. This is described in Attestation 2.

Attestation 2: Proof of attribute distribution (DistAtt)

Input: $\mathcal{D}^{prv}_{tr}$
Output: P, $\mathtt{h}(\mathcal{D}^{prv}_{tr})$, $\mathrm{Sign}_{\mathrm{sk}}(\texttt{'DistAtt'}, \mathtt{P}, \mathtt{h}(\mathcal{D}^{prv}_{tr}))$
Assertion: $\mathtt{P}(\mathcal{D}^{prv}_{tr})$

Attestations for Model Cards

I now cover the various attestations required for verifiable model cards, including: (i) proof of training, which includes attesting the training configuration $\mathcal{T}$ and the model architecture $\mathcal{M}_{ar}$, and (ii) evaluation attestations, which include accuracy, fairness, and robustness on a test dataset $\mathcal{D}^{prv}_{te}$.

Proof of training (PoT) attests that $\mathcal{M}$ was trained on $\mathcal{D}^{prv}_{tr}$. This requires training $\mathcal{M}$ using a training configuration $\mathcal{T}$ on $\mathcal{D}^{prv}_{tr}$. The input to the attestation is $\mathcal{M}_{ar}$, $\mathcal{D}^{prv}_{tr}$, and $\mathcal{T}$. After the training, the attestation output includes the trained model $\mathcal{M}$, the hashes over the inputs $\mathtt{h}(\mathcal{M}_{ar})$, $\mathtt{h}(\mathcal{D}^{prv}_{tr})$, and $\mathtt{h}(\mathcal{T})$, and the signature binding them: $\mathrm{Sign}_{\mathrm{sk}}(\texttt{'PoT'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{M}_{ar}), \mathtt{h}(\mathcal{D}^{prv}_{tr}), \mathtt{h}(\mathcal{T}))$. This is shown in Attestation 3.

Attestation 3: Proof of Training (PoT)

Input: $\mathcal{M}_{ar}$, $\mathcal{D}^{prv}_{tr}$, $\mathcal{T}$
Output: $\mathcal{M}$, $\mathtt{h}(\mathcal{M}_{ar})$, $\mathtt{h}(\mathcal{D}^{prv}_{tr})$, $\mathtt{h}(\mathcal{T})$, $\mathrm{Sign}_{\mathrm{sk}}(\text{'PoT'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{M}_{ar}), \mathtt{h}(\mathcal{D}^{prv}_{tr}), \mathtt{h}(\mathcal{T}))$
Assertion: $\mathcal{M} \leftarrow \mathrm{Train}(\mathcal{D}^{prv}_{tr}, \mathcal{T}, \mathcal{M}_{ar})$

Accuracy attestation (AccAtt) indicates that $\mathcal{M}$ has a certain accuracy ϕ_{acc} as measured using on a specific test dataset $\mathcal{D}^{prv}_{te}$. This ensures (i) there are no false claims about $\mathcal{M}$'s ϕ_{acc}, and (ii) $\mathcal{M}$ was evaluated on (a standardized) $\mathcal{D}^{prv}_{te}$ with no tampering. Laminator, in addition to ϕ_{acc}, generates the hashes of the attestation inputs $\mathtt{h}(\mathcal{M})$ and $\mathtt{h}(\mathcal{D}^{prv}_{te})$, along with a signature binding them, i.e., $Sign_{\mathrm{sk}}(\texttt{'AccAtt'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{D}^{prv}_{te}), \phi_{\mathrm{acc}})$. I summarize this in Attestation 4.

Fairness attestation (FairAtt) extends accuracy attestation to evaluate fairness of $\mathcal{M}$ on $\mathcal{D}^{prv}_{te}$. Fairness can be measured using various metrics such as accuracy parity, and false positive and false negative rates across subgroups. For illustration, I use equalized odds ϕ_{eqodds} as the fairness metric. Attestation 5 shows that $\mathcal{M}$ satisfies ϕ_{eqodds} on $\mathcal{D}^{prv}_{te}$.

Attestation 4: Accuracy attestation (`AccAtt`)

Input: $\mathcal{M}$, $\mathcal{D}_{te}^{prv}$
Output: ϕ_{acc}, $\mathtt{h}(\mathcal{M})$, $\mathtt{h}(\mathcal{D}_{te}^{prv})$, $\text{Sign}_{\text{sk}}(\texttt{'AccAtt'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{D}_{te}^{prv}), \phi_{\text{acc}})$
Assertion: $\phi_{\text{acc}} = \frac{1}{N}\sum_{(x,y)\in\mathcal{D}_{te}^{prv}}^{N} [\![_{\{\mathcal{M}(x)==y)\}}$

Attestation 5: Fairness Attestation (`FairAtt`)

Input: $\mathcal{M}$, $\mathcal{D}_{te}^{prv}$
Output: ϕ_{eqodds}, $\mathtt{h}(\mathcal{M})$, $\mathtt{h}(\mathcal{D}_{te}^{prv})$, $\text{Sign}_{\text{sk}}(\texttt{'FairAtt'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{D}_{te}^{prv}), \phi_{\text{eqodds}})$
Assertion: $\phi_{\text{eqodds}} = |P(\mathcal{M}(X) = 0|S = 0) - P(\mathcal{M}(X) = 0|S = 1)|$ where $(X, Y, S) \in \mathcal{D}_{te}^{prv}$ (assume that datasets have sensitive attributes)

Robustness attestation (`RobAtt`) attests that $\mathcal{M}$ is robust indicated by the *robust accuracy* metric (ϕ_{robacc}) on a test dataset containing adversarial examples ($\mathcal{D}_{te}^{rob}$). This proceeds in two steps: first, I generate $\mathcal{D}_{te}^{rob}$ along with its hash $\mathtt{h}(\mathcal{D}_{te}^{rob})$ inside the enclave. Since $\mathcal{D}_{te}^{rob}$ is generated locally, its hash serves as the certificate during verification. This certifies that $\mathcal{D}_{te}^{rob}$ was created by adding adversarial examples to $\mathcal{D}_{te}^{prv}$, generated using a perturbation budget of ϵ_{rob} using a function `AdvGen`. In the second step, I use accuracy attestation (Attestation 4) but using $\mathcal{D}_{te}^{rob}$ instead of $\mathcal{D}_{te}^{prv}$, yielding ϕ_{robacc}.

Attestation 6: Robustness Attestation (`RobustAtt`)

Generating and Certifying $\mathcal{D}_{te}^{rob}$
Input: $\mathcal{M}$, $\mathcal{D}_{te}^{prv}$, ϕ_{robacc}, ϵ_{rob}
Output:
$\mathcal{D}_{te}^{rob}$, $\mathtt{h}(\mathcal{D}_{te}^{rob})$, $\mathtt{h}(\epsilon_{rob})$, $\mathtt{h}(\phi_{\text{robacc}})$, $\text{Sign}_{\text{sk}}(\texttt{'RobustAttA'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{D}_{te}^{rob}), \phi_{\text{robacc}})$
Assertion: $\mathcal{D}_{te}^{rob} \leftarrow \mathtt{AdvGen}(\mathcal{M}, \mathcal{D}_{te}^{prv}, \phi_{\text{robacc}}, \epsilon_{rob})$
(`AdvGen`: function to generate adversarial examples)

Attesting Robustness
Input: $\mathcal{M}$, $\mathcal{D}_{te}^{prv}$
Output: ϕ_{robacc}, $\mathtt{h}(\mathcal{M})$, $\mathtt{h}(\mathcal{D}_{te}^{rob})$, $\text{Sign}_{\text{sk}}(\texttt{'RobustAttB'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{D}_{te}^{rob}), \phi_{\text{robacc}})$
Assertion: $\phi_{\text{robacc}} = \frac{1}{N}\sum_{(x,y)\in\mathcal{D}_{te}^{rob}}^{N} [\![_{\{\mathcal{M}(x)==y)\}}$

I refer to the attestations described in Attestation 4, 5, and 6 as proof of evaluation.

Attestations for Inference Cards

For verifiable inference cards, I describe the attestation binding input-model-output (also known as proof of inference).

Proof of inference (`PoI`), shown in Attestation 7, attests that the output $\mathcal{O}$ was generated from $\mathcal{M}$ for a given input $\mathcal{I}$. This ensures (i) integrity against a malicious $\mathcal{P}rv$ who can tamper with $\mathcal{O}$, (ii) non-repudiation to prevent $\mathcal{P}rv$ from claiming that $\mathcal{O}$ does not correspond to $\mathcal{I}$, and (iii) prevention of false claims that $\mathcal{O}'$ is from $\mathcal{M}$ for $\mathcal{I}$ instead of $\mathcal{O}$. Note that $\mathcal{O}$ and $\mathcal{I}$ are the inputs and outputs corresponding to $\mathcal{M}$, not the attestation's inputs and outputs.

Attestation 7: Proof of Inference (`PoI`)

Input: $\mathcal{I}, \mathcal{M}$
Output: $\mathcal{O}$, $\mathtt{h}(\mathcal{M})$, $\mathtt{h}(\mathcal{I})$, $\mathrm{Sign}_{\mathrm{sk}}(\texttt{'PoI'}, \mathtt{h}(\mathcal{M}), \mathtt{h}(\mathcal{I}), \mathtt{h}(\mathcal{O}))$
Assertion: $\mathcal{O} = \mathcal{M}(\mathcal{I})$

I describe the various attestations along with the inputs and the outputs in Figure 8.2.

Chain of Attestations and Certificates

Along with the above attestations, certificates from trusted certifiers (e.g., CIFAR) can help $\mathcal{V}er$ to validate the quality of the training or test datasets. These certificates can certify a *non-computational property* such as data provenance or suitability for a particular application. Combining these certificates with attestations can help $\mathcal{V}er$ conclude that $\mathcal{M}$ was trained and evaluated on a dataset considered suitable for a particular task.

The assertions of individual attestations and the external certificates by themselves might not be useful in practice. For instance, proof of inference suggests that *for an input $\mathcal{I}$, $\mathcal{M}$ generates the output $\mathcal{O}$.* However, this information is useful only if $\mathcal{V}er$ knows how good $\mathcal{M}$ is in terms of accuracy, fairness, or robustness. To use attestations in practice, I can chain multiple attestations, enabling broader conclusions to be drawn.

As a motivating example, consider a company testing $\mathcal{M}$ for "good" accuracy on a general test dataset $\mathcal{D}_{te}$ using accuracy attestation (Attestation 4). This helps $\mathcal{V}er$ conclude that their $\mathcal{O}$ is indeed from a good $\mathcal{M}$. Including additional attestations, such as fairness (Attestation 5) and robustness (Attestation 6), demonstrates that $\mathcal{M}$ meets other application-specific requirements. Also, in all cases, certifying $\mathcal{D}_{te}$ ensures that the metrics are computed on a standardized $\mathcal{D}_{te}$ certified by a trusted organization (e.g., CIFAR).

Combining pre-training and in-training attestations (e.g., Attestations 2 and 3) allows one to conclude the following: *$\mathcal{M}$ was trained on $\mathcal{D}_{tr}^{prv}$ satisfying distributional properties P.* However, by combining these attestations with evaluation attestations, one can draw conclusions about the properties of models after training (e.g., accuracy, fairness, and robustness). Hence, one can additionally prove that *$\mathcal{O}$ was generated by $\mathcal{M}$ for a given $\mathcal{I}$, where $\mathcal{M}$ was trained on $\mathcal{D}_{tr}^{prv}$ satisfying property P, and $\mathcal{M}$ satisfies the expected accuracy, fairness, and robustness.*

8.2 Laminator: Framework Design

I present `Laminator`, a framework for generating verifiable ML property cards using ML property attestations from TEEs. This is the first framework for *hardware-based ML property attestation*. `Laminator` establishes bindings between key components in the ML pipeline: (i) the model and its inputs/outputs, (ii) the model and its accuracy with respect to a test dataset, (iii) the model and its training dataset and configuration, and (iv) the distributional properties of a dataset. These bindings are measured (i.e. hashed) and then signed with the TEE's secret attestation key sk, producing *attestations*. Since these can be generated and validated by anyone with the requisite hardware, this approach is inherently scalable and allows verifiers to draw conclusions about the properties of $\mathcal{M}$ and $\mathcal{D}_{tr}^{prv}$. I implement various attestations described in Section 8.1, and revisit completeness and extensions for new attestations in Chapter 9: Section 9.3. I discuss the design and implementation of `Laminator` for $\mathcal{P}rv$ and $\mathcal{V}er$.

Figure 8.1 presents an overview of `Laminator`. A TEE runs a Python script ("measurer") that measures properties of a model, dataset, or inference, and produces an attestation. This attestation is formatted as a property card fragment (i.e., JSON fragment with a model card's structure[2] that can be combined with other fragments to produce the full property card).

I also consider assertions by trusted certifiers (e.g., CIFAR, Hugging Face). These certifiers do not rely on remote attestation but are trusted to provide external certificates. Such certificates may assert that publicly available models or datasets are of high quality, or simply identify a dataset. The combination of (external) certificates and attestations is called the *assertion bundle*. During verification, the assertion bundle is passed to $\mathcal{V}er$ who verifies the attestations using certificates from the TEE manufacturer, and combines them to obtain a set of verified ML property cards.

I implement my proof-of-concept using Intel's SGX as the TEE platform, but the design is not dependent on platform-specific details. It can be equally implemented on any other TEE

[2] https://github.com/huggingface/hub-docs/blob/main/modelcard.md

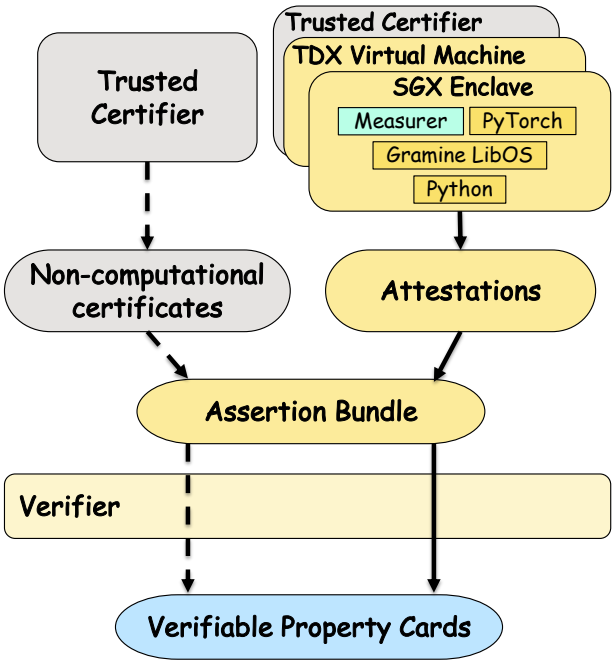


Figure 8.1: **Overview of `Laminator`'s design:** Components already existing or part of the infrastructure are indicated in gray, while components specific to `Laminator` are indicated in yellow. There can be separate enclaves for each attestation type, which are generated by changing the "measurer" (in green). Dashed lines correspond to assertions by a trusted certifier, while solid lines correspond to assertions by a TEE.

with remote attestation support. Different implementations can interoperate as long as $\mathcal{V}er$ can validate each TEE's remote attestations.

Implementation for Prover

$\mathcal{P}rv$ in `Laminator` is a program implemented using Gramine [528], which allows unmodified applications to run inside an SGX enclave. This allows `Laminator` to generate attestations for properties computed with off-the-shelf libraries like PyTorch [529]. I implement the attestations from Section 8.1. Figure 8.2 shows the inputs and outputs from various attestation enclaves, which are describe below:

1. the *dataset enclave* takes $\mathcal{D}_{tr}^{prv}$ as an input and outputs the distribution along with the proof of attribute distribution (Attestation 2).

2. the *training enclave* takes $\mathcal{D}_{tr}^{prv}$, $\mathcal{M}_{ar}$, and $\mathcal{T}$ as input, to train a model inside the enclave, and output the trained model $\mathcal{M}$ along with the proof of training (Attestation 3).
3. the *metric enclave* takes $\mathcal{M}$ and $\mathcal{D}_{te}$ as input, generating metrics for accuracy, fairness, and robustness, along with their attestations as an output (Attestation 4, 5, and 6 respectively).
4. the *inference enclave* runs $\mathcal{M}$ after deployment, producing the model's output and corresponding proof of inference for each input (Attestation 7).

The attestations are generated from a custom ML library built atop PyTorch. A hash of the Python code responsible for these attestations, along with PyTorch and all other files used by the enclave, is embedded into the enclave using Gramine. Gramine provides a library OS that enables the integrity of the Python code to be verified when it is read from disk. This hash is incorporated into the `MRENCLAVE` value included in SGX attestations.

`Laminator` includes a wrapper for Python's `open` library call, which is used to read input files into memory, hashing them at the same time, allowing PyTorch to randomly access measured files without the risk of time-of-check-time-of-use (TOCTOU) vulnerabilities. I add a check for the resulting hash against a manifest[3], enabling a single short measurement to represent a large directory without loading all files into memory.

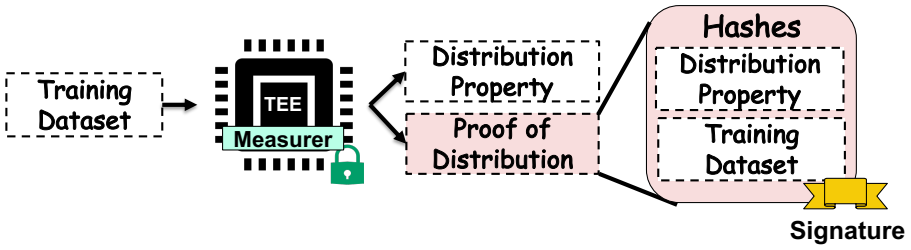
The relevant ML operations are executed inside a secure SGX enclave, producing an output. A hash of the outputs and confidential inputs is computed (also known as a measurement) and embedded in the user data field of an SGX report. The report is then converted into a "DCAP quote", a cryptographically signed attestation artifact proving that the computation ran inside a genuine SGX enclave and that the report contents, were generated by it. Remote parties can verify the quote to confirm both the enclave's authenticity and the integrity of the attested output. These attestation outputs take the form a JSON string representing a fragment of the ML property card.

Implementation for Verifier

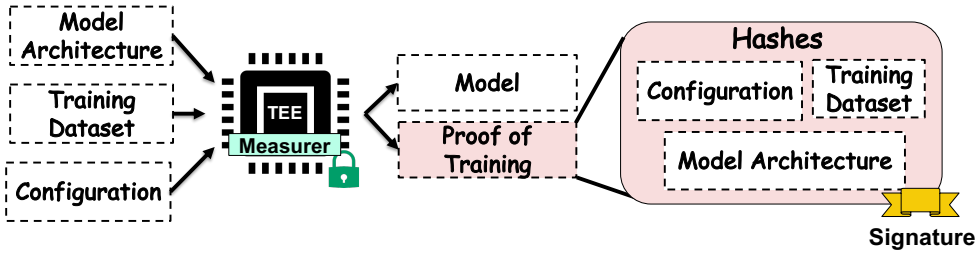
$\mathcal{V}er$ in `Laminator` is a program which takes two main inputs:

- a set of attestations in JSON format, each containing an SGX quote, representing the attestation signature from Section 8.1, and base64-encoded output.
- a set of certificates relating SGX enclave identities to the property that they attest (the property card metadata that they are allowed to assert).

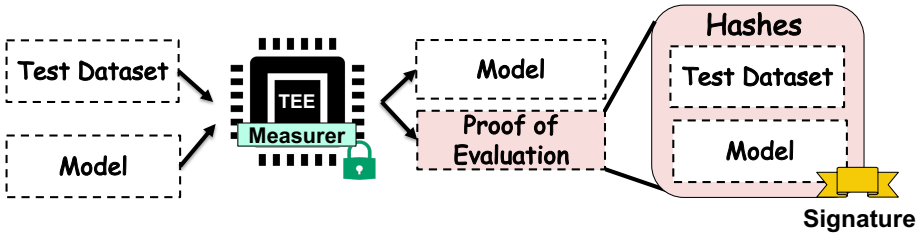
[3]Recall from Chapter 6: Section 6.3 that Gramine's manifest file specifies the security configuration, including which files are accessible and whether integrity checks apply: *trusted files* (hashed at build time and integrity-checked) and *allowed files* (accessible without integrity checks).



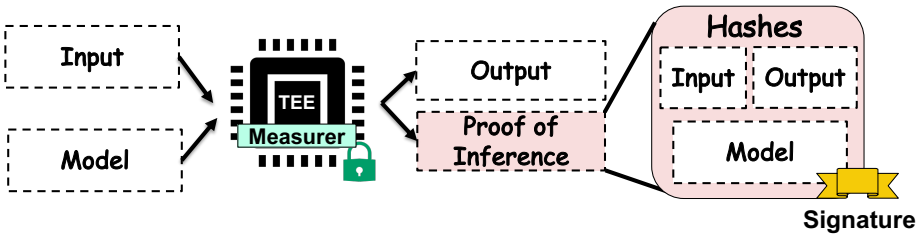
(a) Proof of Attribute Distribution



(b) Proof of Training



(c) Proof of Evaluation



(d) Proof of Inference

Figure 8.2: **Overview of ML property attestations:** The inputs and outputs for various attestations are described along with the content of the attestations.

Since attestations are designed to directly map to YAML-formatted property card metadata, the implementation for $\mathcal{V}er$ is simplified, which needs to only check that the attested JSON corresponds to the property card metadata that the prover is allowed to assert.

$\mathcal{V}er$ first checks the SGX quote to ensure that the attestation was generated by a trusted enclave. It then uses the enclave's identity to retrieve the corresponding rules that define what the enclave is allowed to claim. Finally, it validates the attestation output against these rules by allowing wildcard fields (represented as null), requiring exact matches for structured fields, enforcing that only permitted keys and values are present, and avoiding floating-point values to prevent rounding inconsistencies.

Finally, $\mathcal{V}er$ combines all valid attestations related to $\mathcal{M}$ to construct a property card, which serves as a structured, verifiable summary of $\mathcal{M}$, including its training process, training data properties, and evaluation behavior, similar to standard machine learning documentation. This property card is then stored in a structured format, converted from JSON into YAML (same as the existing model card structures on Hugging Face). Additionally, `Laminator` lets $\mathcal{V}er$ combine TEE attestations with other guarantees (e.g., external certificates in Section 8.1), and assurances from ZKP-like mechanisms.

Authorization

Enclaves should only generate attestations for certified properties, and $\mathcal{V}er$ should not trust attestations from unknown enclaves or those not certified to make a specific assertion. To check for this, an endorser (e.g., Hugging Face) can provide an external certificate for an enclave, in addition to those for datasets or models, to indicate which properties each enclave is allowed to generate. These external certificates can be included in the assertion bundle and validated during the verification, though in my proof-of-concept, they are hardcoded into $\mathcal{V}er$.

8.3 Experimental Setup

I now present the hardware configuration, datasets, model architectures and training configurations, and metrics used for evaluation. Reported results are averages of five runs.

Hardware Configuration

I use a machine with the following configuration:

- **CPU:** Intel Xeon Silver 4514Y.
- **RAM:** 512GB, 8x 64GB (Micron Technology MTC40F2046S1RC48BA1 MHCC), 4800 MHz.
- **Storage:** Micron 7450 MTFDKBG1T9TFR, 1920GB
- **Mainboard:** Supermicro X13SEDW-F

Datasets

I consider three datasets of varying sizes drawn from several domains: a tabular dataset CENSUS [530], an image dataset UTKFACE [531], and a textual dataset IMDB [532]. These datasets are standard benchmarks used in ML literature.

CENSUS contains 48,842 records with 42 attributes from the 1994 US Census, including sensitive attributes such as sex and race. The task is to predict whether an individual's annual income exceeds 50,000 USD. A 50-50 train-test split yields 24,000 records each for training and testing. This translates to 2MB of disk space for the training and test datasets.

UTKFACE is a large-scale face dataset with 48x48 RGB images for classifying individuals as young (<30 years). It includes 11,852 training and 11,853 test data records, with race and sex as sensitive attributes. This translates to 100MB of disk space for the training and test datasets. *IMDB* is a text dataset containing movie reviews from the IMDB website for binary classification of positive and negative sentiments. I use 37,500 training and 12,500 test data records. This translates to 49.5MB and 16.5MB of disk space for the training and test datasets, respectively.

Model Architecture

I train two models on each dataset: a smaller model (with the suffix "-S"), and a larger model (with the suffix "-L"). Table 8.1 summarizes the models and the number of parameters.

CENSUS-S and *CENSUS-L* are multi-layer perceptron (MLP) models with hidden layers of dimensions [128] and [128, 256, 512, 256] respectively, with $\tanh$ activation functions. *CENSUS-S* has $\sim$12K model parameters (0.05 MB) while *CENSUS-L* has $\sim$308K parameters (1.2 MB). I train them for 5 epochs to get an accuracy of 78.33% (*CENSUS-S*) and 81.09% (*CENSUS-L*).

UTKFACE-S and *UTKFACE-L* are VGG11 and VGG16 models [533], respectively. VGG11 with hidden layer dimensions: [64, 'M', 128, 'M', 256, 256, 'M', 512, 512, 'M', 512, 512, 'M'] and VGG16 includes [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'M', 512, 512, 512, 'M', 512, 512, 512, 'M'], where 'M' indicates maxpool. *UTKFACE-S* has $\sim$9.22 million parameters (36.95MB) and *UTKFACE-L* has $\sim$14.72 million parameters (58.96MB). I train the models for 10 epochs to get an accuracy of 80.28% (*UTKFACE-S*) and 80.81% (*UTKFACE-L*).

Table 8.1: **Summary of models across different datasets**

Model	Description	#Parameters	Model Size (MB)
CENSUS-S	MLP: [128]	12,290	0.05
CENSUS-L	MLP: [128, 256, 512, 256]	308,482	1.2
UTKFACE-S	VGG11	9,227,010	36.95
UTKFACE-L	VGG16	14,724,162	58.96
IMDB-S	LSTM: [64, 256, 256]	920,385	3.69
IMDB-L	LSTM: [64, 256, 256, 256, 256]	1,973,057	7.60

IMDB-S and *IMDB-L* are LSTM models with hidden layer dimensions of [64, 256, 256] and [64, 256, 256, 256, 256] respectively. *IMDB-S* has ∼920K parameters (3.69MB) and *IMDB-L* has 1.97 million parameters (7.60MB). I train these models for five epochs to get an accuracy of 85.60% (*IMDB-S*) and 86.18% (*IMDB-L*).

Metrics

The empirical evaluation mainly focuses on efficiency (Section R2), while I discuss other requirements in Section 8.4. I measure efficiency using the execution time. I split the tasks into the following phases of computation:

- *Input loading* is the time spent reading the operation's input data.
- ***Input measurement*** is the time spent hashing the input data to produce a measurement.
- *Data pre-processing* is the time spent on transforming the inputs (e.g., feature selection).
- *Computation* is the time spent on the inference, training, or property measurement.
- ***Output measurement*** is the time spent hashing the output to produce a measurement.
- *Output storage* is the time spent formatting and writing the output data.
- ***Attestation*** is the time spent producing the attestation of the measurements.

`Laminator`'s overheads are marked in **bold** and are the total of ***input measurement***, ***output measurement***, and ***attestation***. I use the total time spent on the remaining phases as the *baseline*, which includes *input loading*, *data pre-processing*, *computation*, and *output storage*; I compare `Laminator`'s overhead against this baseline.

8.4 Evaluation

I present the evaluation of `Laminator` against the requirements in Section 7.1.

8.4.1 R1: Effective

TEE-based attestation enables precise measurement of ML properties because the "measurer" script encodes the target property as a deterministic function within the TEE. The attested result therefore reflects the exact computation and not an approximation. Unlike ML-based mechanisms where correct implementation does not guarantee effectiveness, here a correct implementation indicates effectiveness by accurately measuring the required properties. As long as the script correctly implements the property, hardware-backed attestation ensures the reported value is measured accurately, making the mechanism effective (**R1**).

8.4.2 R2: Efficiency

I assess efficiency by measuring the overhead from measurement and attestation, exploring how it varies with model complexity and dataset size. I aim to validate the following conjectures: the overhead of `Laminator` is

1. determined by a roughly constant attestation overhead, and a measurement overhead dominated by the size of its inputs and outputs. I indicate the inputs and outputs for measurements across the four attestation types below:
   - *Proof of Attribute Distribution:* Training dataset as input, and distribution value as output.
   - *Proof of training:* Training dataset as input, and trained model as output.
   - *Proof of Evaluation (Accuracy Attestation):* Test dataset and trained model as input, and accuracy as output.
   - *Proof of Inference:* Trained model and data record as input, and predictions as output.
2. small in comparison with the computational task being attested for all attestations except for proof of inference, which is discussed later.

Attestations for Datasheets

I measure the execution time of distributional property attestation for verifiable datasheets.

Proof of Attribute Distribution: Table 8.2 summarizes the evaluation of distributional properties for CENSUS and UTKFACE, which include race and sex as sensitive attributes. I do not

Table 8.2: **Performance of distributional property attestation:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). I omit "Output Size" as the output (i.e., distribution value) is a single value of negligible size.

Dataset	CENSUS	UTKFACE
Preprocessing	9.90 s	54.22 s
Input Load	28.89 ms	1.78 s
Computation (Race: z)	73.29 ms	66.28 ms
Computation (Sex: z)	4.38 ms	3.78 ms
Computation (Race: z|y)	26.35 ms	23.49 ms
Computation (Sex: z|y)	10.43 ms	8.74 ms
Output Storage	1.05 ms	1.11 ms
Baseline	10.01 s	54.33 s
Input Meas.	5.90 ms	1.10 s
Input Size	2 MB	100 MB
Output Meas.	0.72 ms	0.87 ms
Attestation	29.85 ms	31.23 ms
Overhead	36.46 ms	1.14 s
Overhead (%)	0.36	2.05

evaluate on IMDB dataset since its text data does not contain any sensitive attributes. I generate attestations for two distributional properties: (i) the distribution of a sensitive attribute z, and (ii) the distribution of z given the classification label y (denoted $z|y$ in Table 8.2).

Attestation time is nearly constant across datasets. As expected, the input measurement time is higher for the larger datasets (UTKFACE). The total overhead incurred by Laminator varies between 0.36% and 2.05%. The size of the output is negligible, so there will be minimal output measurement time incurred (0.72 ms for CENSUS and 0.87 ms for UTKFACE).

Attestations for Model Cards

I report the overhead for attesting two types of ML card properties: proof of training and proof of evaluation (accuracy attestation).

Proof of Training: I train both small and large models on CENSUS, UTKFACE, and IMDB (Table 8.3). The highest recorded overhead was 0.32% for the smallest model and dataset (CENSUS), while for larger datasets (UTKFACE and IMDB), Laminator 's overhead was negligible, at 0.01% or lower. The attestation time does not increase, varying between 15ms and 31ms across each run, while the input measurement and output measurement times increase with the size of the input and output, as conjectured.

Table 8.3: **Performance of proof of training:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange).

Dataset	CENSUS		UTKFACE		IMDB	
Model variant	**-S**	**-L**	**-S**	**-L**	**-S**	**-L**
Preprocessing	9.95 s	9.71 s	54.62 s	54.42 s	70.31 s	70.35 s
Input Load	26.49 ms	28.05 ms	1.80 s	1.83 s	0.33 s	0.35 s
Computation	973.24 ms	0.03 min	310.11 min	643.13 min	1803.16 min	3563.11 min
Output Storage	0.74 ms	0.74 ms	1.81 ms	1.84 ms	0.97 ms	1.09 ms
Baseline	10.95 s	11.53 s	311.05 min	644.06 min	1804.34 min	3564.29 min
Input Meas.	5.86 ms	5.88 ms	1.13 s	1.14 s	96.40 ms	97.35 ms
Input Size (MB)	2.00	2.00	100.00	100.00	49.50	49.50
Output Meas. (ms)	0.23	1.12	24.77	226.79	2.76	5.47
Output Size (MB)	0.05	1.20	36.95	58.96	3.69	7.60
Attestation (ms)	29.16	16.91	25.82	29.78	26.66	19.47
Overhead	35.25 ms	23.91 ms	1.18 s	1.39 s	125.82 ms	122.30 ms
Overhead (%)	0.32	0.21	0.01	0.01	0.00	0.00

Proof of Evaluation (Accuracy Attestation): I measure the accuracy attestation performance for both small and large model variants across all datasets (Table 8.4). The overhead of Laminator is less than 0.35%. The input measurement overhead increases with input size, while the attestation time varies approximately between 22ms and 27ms, consistent with the other attestations. The size of the output is negligible, so the output measurement time is minimal; the overhead is similar for -S and -L models. I note that there is a higher attestation time for -L models compared to -S models for CENSUS and UTKFACE. Since attestation is independent of the model size, I attribute this to some randomness during attestation.

Attestations for Inference Cards

I now report the performance of proof of inference required for verifiable inference cards.

Proof of Inference: I measure the execution time of proof of inference for a single inference averaged over 100 runs (Table 8.5). Since the size of the output is negligible, the output measurement time is very small. Unlike the types of operations I have considered above, the baseline cost for a *single* inference is very small. Consequently, it is no surprise that although Laminator's computation time overhead for the attestation itself remains small (as in previous cases), the proportional overhead is relatively high (between 39% and 3955%). The high overhead stems from the attestation signature being much more costly than inference computation, as shown in Table 8.5.

Table 8.4: **Performance of accuracy attestation:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). I omit "Output Size" as the output (i.e., accuracy) is a single value of negligible size.

Dataset	**CENSUS**		**UTKFACE**		**IMDB**	
Model variant	**-S**	**-L**	**-S**	**-L**	**-S**	**-L**
Preprocessing	9.46 s	10.14 s	54.74 s	55.15 s	70.17 s	70.37 s
Input Load	28.15 ms	34.34 ms	1.96 s	2.08 s	0.36 s	0.37 s
Computation	228.62 ms	306.03 ms	10.45 min	21.38 min	53.78 min	107.46 min
Output Storage	0.95 ms	0.90 ms	1.60 ms	1.51 ms	1.70 ms	1.74 ms
Baseline	9.72 s	10.48 s	11.40 min	22.33 min	54.95 min	108.64 min
Input Meas.	7.03 ms	8.68 ms	1.15 s	1.24 s	0.10 s	0.11 s
Input Size (MB)	2.05	3.20	136.95	158.96	20.19	24.10
Output Meas.	0.01 ms	0.02 ms	0.40 ms	0.32 ms	0.44 ms	0.33 ms
Attestation	26.92 ms	24.15 ms	25.93 ms	22.19 ms	22.61 ms	27.20 ms
Overhead	33.97 ms	32.85 ms	1.18 s	1.27 s	0.13 s	0.14 s
Overhead (%)	0.35	0.31	0.17	0.19	0.00	0.00

Table 8.5: **Performance of proof of inference:** Operations in gray indicate the baseline (orange), while those in blue indicate Laminator's overhead (orange). I omit "Output Size" as the output (i.e., prediction) is a single value of negligible size. I report the overhead computed using attestation ("w/ att") and that with signing using a keypair ("w/ sgn").

Dataset	**CENSUS**		**UTKFACE**		**IMDB**	
Model variant	**-S**	**-L**	**-S**	**-L**	**-S**	**-L**
Input Load	0.25 ms	6.13 ms	221.98 ms	336.42 ms	22.62 ms	41.47 ms
Computation	0.15 ms	0.40 ms	5.79 ms	14.05 ms	29.96 ms	57.97 ms
Output Storage	0.35 ms	1.33 ms	1.62 ms	1.83 ms	1.59 ms	1.61 ms
Baseline	0.75 ms	7.87 ms	229.39 ms	352.29 ms	54.17 ms	101.05 ms
Input Meas.	0.68 ms	2.38 ms	62.98 ms	96.70 ms	5.95 ms	11.97 ms
Input Size (MB)	0.05	1.20	36.95	58.96	3.69	7.60
Output Meas.	0.02 ms	0.40 ms	0.50 ms	0.56 ms	0.46 ms	0.60 ms
Attestation	28.97 ms	26.26 ms	27.38 ms	29.26 ms	23.13 ms	31.78 ms
Overhead w/ att	29.67 ms	29.04 ms	90.86 ms	126.52 ms	29.54 ms	44.35 ms
Overhead w/ att (%)	3955.35	369.18	39.61	55.15	54.52	81.86
Signing in s/w	0.18 ms	3.79 ms	4.40 ms	4.84 ms	4.64 ms	4.56 ms
Overhead w/ sgn	0.88 ms	6.56 ms	67.87 ms	102.10 ms	11.05 ms	17.13 ms
Overhead w/ sgn (%)	1.17	0.83	0.30	0.29	0.20	0.17

I describe how to reduce the per-inference overhead. One approach is to batch multiple inputs from a single client and attest them together, spreading the cost of a single attestation or signature across multiple tasks, resulting in overhead similar to accuracy attestation in Table 8.4. Another approach to reduce cost is to generate a signing keypair during Trusted Application (TA) initialization, perform a single attestation of the public key, and then sign each inference result for indirect, low-cost attestation. I compare the time taken for signing with a keypair versus invoking hardware attestation. I find that the resulting overhead computed using the keypair ("Overhead w/ sgn" in Table 8.4) is less than attestation ("Overhead w/ att").

Summary

These findings highlight `Laminator`'s efficiency, with minimal overhead in comparison with the baseline cost of the computation being attested. The only exception to this is for inference attestation, where the time spent on inference is more comparable to the time spent on attestation; I suggest a mitigation for this by using a signing key pair in software instead of hardware-based attestation. Verification costs are low, involving only certificate checking, and are independent of model size. Hence, `Laminator` is efficient (**R2**).

8.4.3 R3: Scalability

Mainstream TEEs provide attestations based on digital signatures that can be repeatedly verified by multiple verifiers. Additionally, an assertion bundle can include attestations from various provers, which are standardized into a common format. This allows `Laminator` to scale to large numbers of independent provers, without relying on a single entity or TEE type to generate all proofs needed for a verifiable property card. Hence, `Laminator` is scalable (**R3**).

8.4.4 R4: Versatility

`Laminator`'s property definitions are software-based, with hardware only used for isolation and attestation, making it easy to extend to new properties. Provers can write Python scripts to measure new properties, get them certified, while verifiers can validate the claims. Also, new measurer programs (to measure the new properties) can be endorsed without updating all verifiers' software, making it *versatile*. The attestation and certification formats map directly to YAML property card metadata, so $\mathcal{V}er$ requires no modification to support new metrics or ML property cards. Overall, `Laminator` is versatile (**R4**).

8.4.5 R5: Robustness

Laminator does not introduce any cryptographic protocols. Its security guarantees are inherited directly from the underlying attestation mechanism. The TEE is robust due to hardware-enforced isolation, which protects the measurement code and model from host interference. It produces cryptographically signed attestations binding code, model parameters, inputs, and outputs, so any tampering is detectable during verification. This integrity property inherited from TEEs makes the attestations difficult to forge or alter, satisfying robustness (R5).

Attacks on TEEs. Several prior works have demonstrated attacks on TEEs, particularly side-channel attacks, that can compromise confidentiality [534]. Attestation does not introduce new privacy risks, as no additional sensitive data is added to the TEE. Laminator relies on the integrity of the underlying TEE attestation mechanism to provide integrity guarantees for property attestation, complementing the confidentiality properties already offered by the TEE. Hence, leakage of model and training data via side channels is not a concern in this setting. Leakage of attestation keys can compromise the integrity of the attestation mechanism [535, 534], but this is orthogonal to this approach and not affected by Laminator's design. Mitigating such risks through side-channel-resistant code [536], cache partitioning [537], or other techniques [538] is an active area of research by TEE manufacturers. Hence, Laminator relies on the platform developer to ensure integrity, using hardware and software mechanisms to prevent subversion, such as side-channel or extraction of attestation keys.

Takeaway: Overall, Laminator meets all requirements (R1-R5).

8.5 Discussions

Difference from Formal Verification-based Certification: Several works provide "certificates" as accuracy bounds in the presence of adversarial examples (e.g., Li et al. [539]) or for group and individual fairness [540, 541, 542]. These bounds are generated using techniques from formal verification (e.g., randomized smoothing). This is an orthogonal problem because attestation guarantees that a specific computation was executed, not that it is correct. For instance, attestation of training with differential privacy only confirms that differential privacy was used, not whether its implementation or guarantees are correct. These certification techniques can be executed within TEEs, providing both correctness guarantees and a verifiable proof that the techniques were actually carried out by $\mathcal{P}rv$, rather than relying on $\mathcal{P}rv$'s unverified claims of correctness.

Applicability to Other TEEs: The proposed approach is independent of the specific TEE in which the computation is attested and can be extended to TEEs from other vendors (e.g., AMD SEV, Intel TDX, Arm CCA). Where the performance of CPUs is insufficient, hybrid approaches such as Slalom [543] and TEESlice [544] enable a CPU-based TEE to offload model layers to a GPU, using cryptography to ensure data confidentiality and output integrity. This speeds up computation compared to CPU-only execution, but the added cryptographic overhead and data transfer between CPU and GPU reduce performance compared to standard GPUs. New TEE-enabled GPUs (e.g., NVIDIA's H100 [524]) accelerate computations, allowing evaluation of larger models (e.g., LLMs) with comparable performance [545, 546].

Relation to Part-1 (Unintended Interactions in ML): The proposed attestations can be extended to demonstrate whether practitioners have accounted for the unintended interactions identified in **Part-1**: Chapters 3, 4, and 5. For Chapter 3, proof of evaluation can be extended to attest that the proposed guideline was applied to predict whether an unrelated risk will increase or decrease when a defense was applied. Additionally, practitioners may decide to empirically evaluate the attacks corresponding to the risks predicted to increase. The evaluation against those attacks and the corresponding attack success rates can be included as part of proof of evaluation. For Chapter 4, attestations can be adapted according to the stage at which a defense is applied. Pre-training defenses that modify the training dataset can be attested through "proof of dataset transformation", similar to proofs of attribute distribution. In-training defenses can be incorporated into proof of training by demonstrating their inclusion in the optimization objective. Post-training defenses, such as input or output filtering and transformations, can be incorporated into proof of inference to attest their deployment as well as binding the outputs to the defense for specific inputs. For Chapter 5, attestations can demonstrate that the proposed guideline for assessing collusion potential was applied and can provide evidence that deployed defenses reduce the likelihood of collusion.

Contribution in Support of the Thesis Statement

ML property cards, widely used by companies such as Hugging Face, should be verifiable to prevent manipulation by malicious provers (e.g., model trainers). I propose `Laminator`, the first framework to generate verifiable ML property cards using hardware-based attestations, and demonstrate that `Laminator` is effective, efficient, scalable, versatile, and robust. Through `Laminator`, I successfully answer **RQ2.2** to support the thesis statement.

Chapter 9

Summary and Future Directions

I conclude with a summary of the thesis contributions (Section 9.1), lessons learned (Section 9.2), future directions (Section 9.3), and situate the contributions within the broader research context (Sections 9.4).

9.1 Summary of Contributions

In this thesis, I focused on two deployment concerns: unintended interactions in ML (**Part-1**), and accountability in ML pipelines (**Part-2**). I summarize the contributions towards answering the research questions to support the thesis statements for each part.

Part-1: Unintended Interactions in ML

In **Part-1**, recall that my goal was to defend the following thesis statement:

> **Thesis Statement**
> Unintended interactions can be systematically understood and predicted by identifying the underlying factors that influence the effectiveness of defenses and susceptibility to risks.

I summarize my contributions in support of the thesis statement by answering the following three research questions for the three types of unintended interactions:

- RQ1.1: ***What are the factors underlying unintended interactions among defenses and unrelated risks, and how can we effectively predict unexplored ones?*** I identified overfitting and memorization as the primary causes underlying unintended interactions among defenses and unrelated risks. I presented a framework that includes factors influencing overfitting and memorization covering the characteristics of the (i) training data (e.g., size, distribution), (ii) objective function (e.g., distance to boundary, loss curvature), and (iii) model (e.g., capacity). I showed that this framework helps explain known interactions, and uncover unexplored ones. Using this framework, I proposed a guideline for conjecturing about unexplored interactions by examining how a defense changes a factor and how those changes correlate with susceptibility to a risk. I conjectured the nature of three previously unexplored interactions, and empirically validated them.
- RQ1.2: ***How can we effectively combine ML defenses while avoiding conflicts?*** I identified several factors underlying conflicting interactions among ML defenses, based on (i) where each defense is applied in the ML pipeline (pre-, in-, or post-training), and (ii) whether their underlying mechanisms interfere (e.g., a later defense minimizes a risk leveraged by an earlier defense). By accounting for these factors, I proposed a combination technique, `Def\Con`, and showed that it can accurately predict whether two defenses will conflict, thereby determining whether they can be effectively combined.
- RQ1.3: ***What are the underlying factors that aid the potential for collusion among adversaries, and how can we predict the potential for collusion?*** To study the potential for collusion among adversaries, where one adversary can execute an attack to amplify others, I presented a framework that captures the key enabling factors. For collusion between train-time and inference-time adversaries, I showed that train-time attacks can influence overfitting and memorization, thereby increasing the effectiveness of various inference-time attacks. Here, the framework includes factors related to overfitting and memorization that an adversary can manipulate. For collusion among inference-time adversaries, I showed that the knowledge inferred as outcome of one attack can satisfy the prerequisites of another. Here, the framework includes each attack's prerequisites and outcomes as factors. Using this framework, I studied the potential for collusion and proposed a guideline to explain collusion in prior work, and predict the potential for unexplored cases of collusion.

Overall, I identified and explored three types of unintended interactions in ML by answering their corresponding research questions, showing that identifying and analyzing the underlying factors enables both the explanation of existing interactions and the prediction of unexplored ones. In doing so, I successfully defended the thesis statement.

Part-2: Accountability in ML Pipelines

In Part-2, recall that my goal was to defend the following thesis statement:

> **Thesis Statement**
> Ensuring accountability of ML operations by designing technical mechanisms that are effective, efficient, versatile, scalable, and robust, is feasible.

I summarize my contributions in support of the thesis statement by answering the following two research questions:

- RQ2.1: ***Can software-based ML property attestation mechanisms meet all requirements?*** Using the example of proof of attribute distribution, I explored two software-based mechanisms: an ML-based mechanism that repurposed distribution inference attacks, and a cryptographic mechanism using MPC. I showed that the ML-based mechanism was neither effective nor robust, while the cryptographic mechanism was inefficient. I then proposed a hybrid mechanism that combined both to improve efficiency and effectiveness; however, I showed that it was not scalable to multiple provers and remained impractical. I thus highlighted the need for alternative mechanisms that meet all requirements.
- RQ2.2: ***Can hardware-based ML property attestation mechanisms address the limitations of software-based ones, and meet all requirements?*** Inspired by remote attestation in TEEs, I presented `Laminator`, a framework that leveraged hardware-based TEEs to provide ML property attestations, including proof of attribute distribution, training, evaluation, and inference. I showed that `Laminator` accurately measures the properties of interest (effective), incurs low overhead compared to standard ML operations (efficient), scales to multiple provers and verifiers (scalable), supports a wide range of ML properties (versatile), and resists malicious provers attempting to evade attestation (robust).

Overall, I explored mechanisms for attesting ML properties and showed that hardware-based mechanisms using TEEs are a promising direction, as they are effective, efficient, scalable, versatile, and robust. In doing so, I successfully defended the thesis statement. Furthermore, these attestations can demonstrate that practitioners applied the claimed defenses, as well as evaluated and accounted for the unintended interactions identified in Part-1.

9.2 Lessons Learned

Part-1: Unintended Interactions in ML

Limits to Overfitting/Memorization Hypothesis: I identified overfitting and memorization as plausible underlying causes because prior work consistently links the effectiveness of many attacks and corresponding defenses, to these factors (Chapter 3: Section 3.2.3; Chapter 5: Section 5.2). This connection is particularly evident for classifiers, but has also been extended to generative models. For example, risks such as adversarial-prompt jailbreaks and privacy attacks are often associated with memorization and overfitting [72, 86, 87].

However, overfitting and memorization do not explain all risks across all models and deployment settings. In agentic systems, for instance, risks such as prompt injection [547, 548, 549, 550], and privacy violations arising from inappropriate information flows, are not captured by either overfitting and memorization [551, 552]. Prompt injection attacks stem from a model's inability to distinguish between data and instructions (similar to memory corruption vulnerabilities in systems security [553]), whereas inappropriate information disclosure reflects a failure to determine what information should be shared in a given context. These risks therefore arise from different underlying causes.

Consequently, overfitting and memorization provide a reasonable basis for relating attacks and defenses at the level of an individual model, as considered in Chapters 3 and 5. However, when analyzing system-level risks, these factors alone are insufficient. Additional system-level factors must be explored to explore risks, defenses, and their interactions.

Advice to Practitioners: Practitioners should update their pipelines to evaluate and account for unintended interactions. After training and before deployment, practitioners should consider the deployment ecosystem, including interactions with other system components and potential failure modes. This includes conducting comprehensive risk evaluations as follows:

- Evaluate defenses for unintended interactions with unrelated risks using the guideline in Chapter 3. Where resources allow, empirically validate whether predicted risk increases and refine $\mathcal{M}$ to mitigate them. Alternatively, certain factors can help anticipate and mitigate risks and unintended interactions without extensive empirical evaluation. For example, long-tailed datasets (e.g., in healthcare) may increase susceptibility to multiple risks (Table 3.2), and such datasets can be transformed (e.g., removing long-tail outliers) to minimize the risks.
- When combining defenses against multiple risks, identify candidate defenses and apply `Def\Con` to detect conflicts (Chapter 4). If conflicts arise, adjust the ordering of defenses or select alternative variants, iterating until `Def\Con` reports no conflicts. Finally, evaluate

whether certain defenses also mitigate unrelated risks (e.g., differential privacy; Table 3.4), potentially reducing the need to apply additional defenses.
- Evaluate the potential for collusion among adversaries using the guideline from Chapter 5, and assess how the above design choices affect this potential. Revisit and refine model design decisions based on the results.
- In Chapter 3, I treat susceptibility to all risks equally. In practice, however, a 5% increase in one risk may be less acceptable than a 10% increase for another. Such distinctions are not captured because evaluation of risks rely on different metrics. Similarly, in Chapter 4, `Def\Con` reports only alignment and conflict between defenses, making it difficult to compare the magnitude of gains and losses across different defenses. For example, a 5% improvement against one risk may be more valuable than a 10% improvement against another. However, there is no common metric to support such comparisons. In both cases, the tolerance for risk is application-dependent and must be determined by practitioners. Practitioners can map changes in risk susceptibility or defense effectiveness to a common metric, such as monetary cost, enabling more consistent evaluation and comparison across risks and defenses.

Part-2: Accountability in ML Pipelines

In **Part-2**, I empirically demonstrate the limitations of both ML-based and cryptographic mechanisms to attesting ML operations. I will now discuss whether software-based attestation mechanism are fundamentally limited to attest ML properties.

- For ML-based mechanisms, existing work largely re-purposes privacy attacks (e.g., `MemInf`, `DistInf`, and `AttInf`). While these attacks only need to achieve better-than-random performance to demonstrate privacy leakage, auditing requires a substantially stronger guarantee. An ideal auditor would effectively have oracle access to the training distribution, enabling perfect true positive rate with zero false positives. In practice, however, these attacks rely on limited model observables, such as gradients or predictions, which may not contain sufficient information to support highly accurate attestation. Consequently, there may exist a fundamental upper bound on attestation accuracy achievable from such observables, which may fall short of the attestation requirement. This is evident from high false-positive rates observed in practice (Chapter 7 and [520, 521]), further reducing their suitability as attestation mechanisms. The formal proof for this is left as future work.
- On the other hand, modern ML models rely heavily on non-linear operations, which are expensive to support using cryptographic mechanisms. Although significant progress has been made in improving the efficiency of techniques such as 2PC and ZKP for ML training and inference [514], the underlying mismatch between cryptographic computation and non-linear operations required for ML models, may limit scalability to large models. Furthermore,

model developers may be unwilling to modify standard ML architectures and operations solely to improve cryptographic efficiency, further limiting their practicality.

Given the above limitations of software-based mechanisms, I argue that hardware-based attestations using TEEs are more promising to efficiently furnishing various ML property attestations in the current ML ecosystem.

9.3 Future Research Directions

I will now present future research directions specific to each chapter (Section 9.3.1), followed by broader research directions for each part of the thesis (Section 9.3.2).

9.3.1 Chapter-wise Future Directions

Common Future Directions for Chapters 3, 4, and 5

- **Generalization beyond Classifiers:** I focused on classifiers because their attacks and defenses are well established and extensively benchmarked in the literature. I will discuss how the guidelines and frameworks can be extended to generative models.
  - For unintended interactions among defenses and unrelated risks (Chapter 3), many factors related to overfitting and memorization, also apply to generative models. Although the definitions of overfitting and memorization differ in generative settings [70, 89], generative models introduce additional factors (e.g., token sampling distribution), while some classifier-specific factors (e.g., decision-boundary distance) are no longer relevant. By identifying these factors and their relationships with risks and defenses, the same guidelines can be extended to generative models.
  - For combining ML defenses (Chapter 5), the interference factors I consider (e.g., the placement of defenses and whether their mechanisms conflict) are sufficiently general to apply to generative models. For example, for fine-tuning generative models on specific task overrides safety guardrails [554], `Def\Con` would suggest that safety alignment should be applied as the final training step to avoid conflict. The type of interference may differ for generative models requiring extending `Def\Con` to cover those additional factors.
  - For collusion potential (Chapter 5), the evaluation combines empirical analysis with evidence from prior work. I already consider studies involving generative models and find that the identified factors help explain collusion potential in such models, suggesting that these factors generalize beyond classifiers. While additional factors specific to generative

models may exist, I excluded them due to insufficient correlations with attacks. Such factors can be incorporated in future extensions given sufficient correlations with attacks.
  - Finally, for ML property attestations (Chapter 8), `Laminator` served as a proof of concept for classifiers. Since then, my co-authors and I have proposed a framework, PAL*M that extends attestation to generative models running on H100 GPUs with trusted computing, demonstrating that efficient attestation is also feasible for generative models [546].
- **Generalization beyond Images:** The frameworks identify fundamental factors that generalize across modalities. Overfitting and memorization are fundamental properties of ML models, and I assumed that risks and defenses can be related to them regardless of modality. For a given task (e.g., classification or generation), factors for overfitting and memorization remain the same, although their measurement differs [70, 89]. In other words, different datasets may exhibit different risk profiles and unintended interactions (e.g., datasets with long-tailed distributions may be more susceptible to certain risks than others). Consequently, the same classes of factors should apply across dataset modalities, with any modality-specific factors incorporated as extensions to the respective frameworks. For Chapter 4, since `Def\Con` 's steps are modality-independent, I conjecture that it can be applied similarly to other data types. I validated this using classifiers on a tabular dataset (i.e., CENSUS) and found that `Def\Con` generalizes beyond the specific modality studied, suggesting broader applicability across modalities. For Chapter 5, as part of the systematization, I considered prior work covering empirical evaluation of different modalities and showed that the current factors can successfully explain the potential for collusion. This suggests that the framework and its factors apply across modalities.
  In case there are additional modality specific factors, those can be easily incorporated in the frameworks, followed by identifying the correlations with risks and defenses (Chapters 3 and 5) or relation with conflicts (Chapter 4), and applying the corresponding guideline. Finally, analyzing dataset characteristics can therefore help predict changes in risk susceptibility, and defense effectiveness applied to datasets to update these factors to suppress unintended increase in risks or collusion potential.
- **Causality and Theoretical Guarantees:** The frameworks in Chapters 3 and 5 focus on correlations rather than causation, since the limited interpretability of model behavior and the poor understanding of factor-attack dynamics make it difficult to causally identify enabling factors. As a result, the framework is conceptual and the proposed guideline is heuristic in nature. The framework and guideline identify correlations rather than causal relationships. Prior work has established causal links between susceptibility to membership inference risk and several underlying factors (e.g., model capacity and training data size) [555]. A similar framework using Bayesian networks could be developed where *causal reasoning* can determine causal relationships between different factors, defenses, and risks. This framework could help identify load bearing factors from this analysis (i.e., whether all the identified

factors in the frameworks are redundant).
Since both the guidelines from Chapter 3 and 5, and `Def\Con` (Chapter 4) define classification rules, their alignment and conflict conditions can be encoded as a formal relation in an SMT solver (e.g., Z3, CVC5) or a proof assistant (e.g., Lean 4, Rocq/Coq, Isabelle). One can then prove meta-properties independent of empirical validity, such as *determinism* (every input receives exactly one label), *totality* (all interactions/combinations/collusions are covered), and *consistency* (no input is simultaneously classified as two separate classes). Formal verification can assess the soundness of guidelines and provide counterexamples whenever the rules are incomplete or contradictory.
Finally, theoretical frameworks can be developed to characterize (a) fundamental trade-offs between defenses and risks (e.g., conditions on training dataset size under which adversarial training increases membership inference [59]), (b) inherent conflicts between pairs of defenses, and (c) potential for adversarial collusion.

Chapter 3: Unintended Interactions among ML Defenses and Risks

- **Applying Guideline to Exceptions:** There were several interactions where the guideline was not applicable due to no common factors among defenses and risks. A more comprehensive evaluation of how factors correlate with effectiveness of defenses and susceptibility to risks is needed to apply the guideline to such interactions (i.e., expanding the empty columns for factors in Table 3.2).
- **Including Threat Models:** The guideline should be extended to account for exceptions arising from differing threat models. One approach is to first identify the threat model and depending on whether $\mathcal{A}dv$ has whitebox or blackbox access to $\mathcal{M}$, then either apply the proposed guideline as is, or adjust its recommendation.
- **Fairness Metrics:** Various fairness metrics (such as equalized odds, equal opportunity) are mutually incompatible and involve inherent trade-offs [556]. Hence, future work can explore how optimizing for different fairness metrics impact nature of interactions.
- **More Effective Attacks/Defenses:** Stronger attacks or more effective defenses are likely to produce stronger correlations with the factors identified in Table 3.2. Thus, while the overall nature of the interactions will remain unchanged, I speculate that the magnitude of increase or decrease in unrelated risks will depend on the effectiveness of the attacks or defenses. Future work can empirically validate this conjecture.

Chapter 4: Combining ML Defenses without Conflicts

- **Learning-based Component for `Def\Con`:** While it is possible to train a model (e.g., a decision tree) to predict the nature of interactions for a given combination, this would require substantially more training data (e.g., features covering a wide range of defenses and their combinations). This exceeds what can be obtained from the eight combinations in prior work.

For the current set of defenses, the heuristic guideline is sufficient to achieve reasonable accuracy. Incorporating a learning-based component is left for future work.

- **Evaluating Omitted Defense Combinations:** I *speculate* the omitted defense combinations from Section 4.4.2: `EvsnRob-Pre`, `DiffPriv-Pre`, `GpFair-Pre`, and `GpFair-Post`. Since `EvsnRob-Pre` targets adversarial examples and makes local changes to $\mathcal{D}_{tr}$, I expect its combination with other defenses to behave similar to `MdlWM-Pre` and `DtWM-Pre`. Additionally, `DiffPriv-Pre` and `GpFair-Pre` make global changes by transforming all data records in $\mathcal{D}_{tr}$ and should be applied before other defenses, as I expect them to avoid conflicts. `GpFair-Post` makes global changes in post-training stage to all predictions, and the behavior is likely to be similar to `PoisnRob-Post` which also makes global changes to $\mathcal{M}$ in post-training stage. Validating these interactions is left for future work (Chapter 9: Section 9.3.1). For defenses omitted due to (i) poor effectiveness: further work is required to design more effective defense variants (e.g., pre-training or post-training evasion robustness). (ii) incompatible datasets: set up experiments for the specific dataset where the defenses work well (e.g., tabular datasets instead of image datasets) and evaluate their combinations. These can then be combined with other defenses and evaluated.
- **Combination-friendly Defenses:** An important direction for future work is the design of defenses that can be seamlessly combined with others. Such combination-friendly defenses would enable practitioners to select defenses based on application requirements and system constraints without relying on ad hoc optimizations. Developing these defenses, however, introduces new design challenges that remain to be explored.
- **Accounting for Privacy Budget ϵ_{dp} in `DiffPriv`:** Currently, combining defenses with `DiffPriv` does not incur additional ϵ_{dp} consumption due to the post-processing property; ϵ_{dp} remains unchanged after composition. However, other defense combinations may require additional privacy accounting. An important direction for future work is developing methods to account for ϵ_{dp} consumption when combining `DiffPriv` with defenses.
- **Other Factors Underlying Conflicts:** While I identify two possible factors underlying conflicts among defenses, I do not claim this to be exhaustive. There may be other contributing factors that could be incorporated into `Def\Con` to improve its accuracy. One possible factor is the choice of different l_p-norm distances used by some defenses (e.g., `EvsnRob-In`, and adversarial example-based `MdlWM-Pre` and `DtWM-Pre`). Prior work has shown that objectives targeting robustness under different l_p-norm bounds can be conflicting [557]. When combining defenses, Thakkar et al. [373] show that optimizing watermarking can resolve conflicts with adversarial training. I leave the identification of additional factors as future work.
- **Defense aware Adversaries:** `Def\Con` currently assumes that individual defenses are robust and that their robustness is preserved when combined. However, I did not evaluate adversaries that are explicitly optimizing their attacks against the combined defense pipeline. Such attacks may uncover vulnerabilities that are not present when defenses are evaluated

individually. Designing such attacks, and developing defenses that anticipate and withstand them while ensuring effective combination, is left for future work.

- **Unintended Interactions with Unrelated Risks:** I evaluate interactions only for the targeted risks addressed by the combined defenses. However, combining defenses may also increase or decrease susceptibility to other unrelated risks. Evaluating and exploring unintended interactions for defense combinations is left for future work.

Chapters 7 and 8: ML Property Attestation Mechanisms

- **Extending to Consumer Devices:** Hardware-based attestations are practical in today's ML ecosystem due to the availability of GPU-based TEEs on major cloud platforms such as Amazon, Microsoft, and Google. Some providers, such as Amazon, also offer software support. However, consumer devices often lack TEEs and instead include trusted platform modules (TPMs). Unlike TEEs, where the measurement code executes in a protected environment and cannot be tampered with by an adversary, the adversary can access and potentially manipulate the measurement script for TPMs. Consequently, additional mechanisms are required to verify the integrity of the measurement process before attestations can be trusted. This is left as future work.
- **Extending to Other Attestation Types:** I illustrate three types of ML property cards and four types of model attestations in Section 8.1, but I do not claim completeness. There could be other attestation types and ML property cards but `Laminator` can be extended to account for them by modifying the "measurer" script by $\mathcal{P}rv$ (Section 8.2). Any property that can be expressed in Python code can be attested using Laminator. For instance, `Laminator` can be extended to include other attestation types, such as ecosystem cards, a combination of multiple models trained on different datasets, using verifiable model cards and datasheets [558]. Also, I can attest various steps in DPSGD training within `Laminator` to demonstrate to $\mathcal{V}er$ that $\mathcal{P}rv$ has used the algorithm. Model explanations (during inference) would be handled similarly to proof of inference. While I focus on static properties for various ML operations, there are other runtime properties such as the environmental impact of running ML models in GPUs (e.g., carbon emissions). These attestations are left for future work.

9.3.2 Broader Future Directions

I now discuss broader research directions related to unintended interactions in ML (**Part-1**) and accountability in ML pipelines (**Part-2**). Ongoing innovations enable *agents* based on large language models (LLMs), to access tools, APIs, and web resources to perform complex tasks on behalf of users. The deployment of these LLMs in agentic settings raises various risks, which may be amplified in multi-agent interactions. Consequently, studying deployment concerns

for LLMs, beyond those explored in Part-1 and Part-2, is important for ensuring their trustworthy deployment. I outline potential research directions within each part for such LLMs.

Unintended Interactions in Language Models (for Part-1): I discuss additional examples of unintended interactions for generative models and agents:

- **Unintended Interactions among LLM Optimizations and Risks:** LLMs can be optimized for various objectives, including efficiency (e.g., pruning, quantization, knowledge distillation), privacy (e.g., DPSGD), utility (e.g., model editing), and safety (e.g., alignment). At the same time, they remain vulnerable to various risks such as jailbreaks, discriminatory behavior, PII leakage, and data reconstruction. Applying different optimizations can influence an LLM's susceptibility to such risks, giving rise to unintended interactions. Existing works have explored limited interactions. For example, standard fine-tuning can increase misalignment [559, 554] and increase bias [560]; while quantization and model steering can increase susceptibility to jailbreaks and degrade utility [561, 562, 563]. However, a systematic evaluation across various optimizations, risks, and the underlying factors is still lacking.
  Both optimizations and risks are influenced by shared underlying factors, such as memorization. In Chapter 3, I analyzed unintended interactions in classifiers, identifying overfitting and memorization as the underlying causes. However, this framework does not directly extend to LLMs: some factors (e.g., distance of training data points to the decision boundary) are not applicable, while LLM-specific factors (e.g., token sampling) must be incorporated. Future work can systematically study how effective optimizations influence susceptibility to different risks, and to identify the factors underlying these unintended interactions, and understand their impact.
- **Emergent Misalignment:** Fine-tuning on a specific task can induce broad misalignment (or harmful behavior) in other contexts (aka emergent misalignment) [559]. This parallels Chapter 3, where optimizing specifically for a defense can influence the susceptibility to unrelated risks. Here, however, the optimization is for general tasks (not just defenses), and the degradation has a broader scope (may affect arbitrary tasks). The first challenge is to systematically identify which capabilities are harmed (an objective which differs from Chapter 4). Following this, identifying the mechanisms driving these unintended interactions can inform the design of mitigations. As in Chapter 3, overfitting and memorization are likely contributors, but the specific factors differ between classifiers and generative models. Addressing this requires revisiting the framework from Chapter 3 to identify the factors relevant to generative models (and not classifiers), to better understand emergent misalignment, and develop methods to prevent it.
- **Conflicting Optimizations in LLMs:** Similar to emergent misalignment, it has been observed that task-specific fine-tuning can degrade or remove existing defenses (e.g., safety mechanisms) [554]. This parallels the conflicts among defenses discussed in Chapter 4,

highlighting the need for methods that can effectively combine multiple defenses in generative models and agents. This problem is particularly pronounced in open-weight LLMs, where users may attempt to remove safety protections such as misalignment safeguards. A key direction is developing methods that resist modification of existing defenses during task-specific fine-tuning. Some work on tamper-resistant fine-tuning aims to preserve core functionality [564, 565], but existing schemes are not yet robust [566].

- **Robust Alignment despite Conflicting Values:** Aligning agents with human values and norms is inherently a sociotechnical problem. Similar to combining multiple defenses against different risks (Chapter 4), practitioners aim to align models with multiple objectives that reflect human expectations. However, these expectations can conflict due to their dynamic, context-dependent nature, making the alignment objective challenging. Such conflicts can be viewed as unintended interactions arising from optimizing multiple objectives simultaneously. Designing algorithms to resolve these conflicts is essential for enabling coherent and justifiable decisions. An interesting direction is to explore social choice theory [567], game theory, and mechanism design to develop practical approaches for aligning agent behavior.

Accountability in Broader ML Ecosystem (for Part-2): Revisiting system and network security principles is crucial to prevent agents from tampering with traditional Internet infrastructure (e.g., databases), enable verifiable execution of agents, enforce accountability across the broader AI ecosystem, and support demonstrable compliance with regulations.

- **Verifiable Agents:** There is a need for provable guarantees of agent behavior. Although formal verification is challenging for natural language, it is feasible in constrained settings such as verifiable model routers (which select the appropriate model for a given prompt) and agent planners (which predict the correct tool invocation for formally specified tasks). Inspired by control-flow and data-flow integrity, a promising direction is *runtime attestation*, in which each agent tool invocation passes through a trusted "trampoline function" that measures and attests to execution traces using TEEs. This can help mitigate prompt injection attacks that induce unauthorized actions.
- **Ecosystem Property Attestations:** ML workloads, especially generative models, form a complex ecosystem of diverse assets (e.g., models, datasets, and applications across organizations) that can be represented as a graph, with assets as nodes [558]. Existing attestations for individual models or local ecosystems fail to capture cross-organizational dependencies. A global ecosystem manager (e.g., Hugging Face) can construct a unified graph, but the manager is untrusted. Thus, the graph construction must be done verifiably and there is a need to design corresponding mechanisms for constructing *verifiable ecosystem graphs*. One possible direction is to integrate attestations from local managers and `Laminator` to establish trustworthy links, enabling efficient reasoning about relationships without costly manual inspection. For example, a proof of training can create a verifiable link between a dataset

and a model either using `Laminator` or attestations from other sources (e.g., ZKPs).

- **Regulatory Sandboxes:** The EU's AI Act [488] requires that "member states shall ensure that their competent authorities establish at least one AI regulatory sandbox at national level" (Art. 57(1)). These sandboxes are intended to facilitate development, training, testing, and validation under regulatory supervision, and participation may help providers demonstrate compliance with the AI Act. The design of such sandboxes remains underexplored. One possible approach is to combine ML property attestations for transparency and compliance verification with hardware-based solutions such as TEEs to enforce isolation. Additional mechanisms can further control information flows arising from model interactions, such as those in agents or chatbots.

9.4 Concluding Remarks

I conclude by situating this thesis and its contributions within a broader research context. Most literature on "trustworthy ML systems" focuses on identifying and mitigating risks within specific pillars of trust, namely security, privacy, fairness, transparency, and safety. This includes designing attacks that exploit vulnerabilities within a given pillar and corresponding defenses to mitigate them. While this line of work is important, it is not sufficient to ensure *trustworthy deployment* of ML models. As highlighted in Figure 9.1, I highlight the need to go beyond exploring isolated risks and defenses, and consider broader deployment concerns:

- **Holistic System Analysis:** ML models typically operate as components within larger systems and interact with traditional Internet infrastructure, external APIs, and tools. This requires a system-level analysis that accounts for trade-offs and unintended consequences of design choices when integrating ML models into larger systems. Examples of problems within holistic system analysis include privacy side channels [568], and unintended interactions (in **Part-1**), exploiting memory corruption vulnerabilities in ML libraries or model conversion tools [569, 570, 571, 572], side-channel attacks [451], and hardware fault injection attacks [456]. A holistic system analysis is essential because simply adding more ML-specific defenses is insufficient in many cases as their robustness is limited [126, 127]. Many risks are at a system-level (e.g., sensitive information leakage [551] or indirect prompt injection [548]), and must be addressed by drawing on systems security research, including control- and data-flow integrity, attestation, contextual integrity for privacy [552], trusted hardware, and cryptographic primitives.
- **Governance of ML Systems:** Despite designing effective defenses against individual risks and accounting for holistic system analysis, practitioners must be able to demonstrate these efforts in a verifiable manner (e.g., to regulators). Several regulations across jurisdictions

and guidelines from frontier labs (e.g., Google [573], Anthropic [574], and OpenAI [575]) mandate accountability, transparency in the ML training pipeline, and demonstrable compliance [484, 485, 486, 487, 488, 489, 490, 491]. However, these do not spell out concrete technical mechanisms for ensuring compliance, making their design an important open problem. Within the broader theme of ML governance, I focus on the specific problem of enabling accountability in ML pipelines (in **Part-2**). Additional challenges include reconciling the expectations of legal scholars and regulators with the technical design of mechanisms, and aligning models with legal statutes.

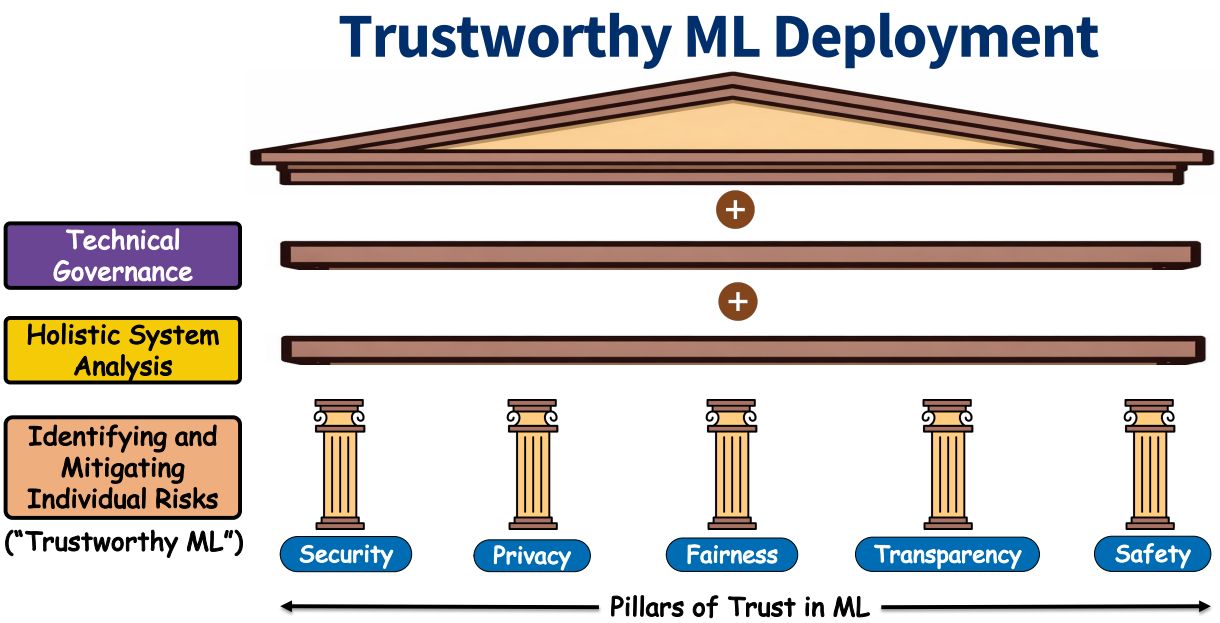


Figure 9.1: In addition to identifying and mitigating risks within individual pillars of trust, trustworthy ML deployment requires a holistic systems analysis (e.g., unintended interactions), and supporting governance, (e.g., for accountability and regulatory compliance).

Need for Systems-Security Mindset: Since the first demonstrations of adversarial examples over a decade ago [100], numerous defenses have been proposed within the individual pillars of trust. However, many defenses for ML classifiers have been repeatedly bypassed [126, 127, 576], a pattern now emerging in generative models [577, 566, 547]. This highlights not only technical limitations but also a deeper misalignment between how the community conceptualizes defense design and the actual susceptibility to attacks. I argue that this misalignment stems from the lack of a systems security mindset in the ML community when designing defenses. There is a strong need to adopt such a mindset, which involves the following:

1. **Precise Threat Modeling to Design Strong Attacks:** Precise and explicit threat modeling is the first step toward robust defenses. It requires defining $\mathcal{A}dv$'s role, capabilities, and access to $\mathcal{M}$ and $\mathcal{D}_{tr}$ (Chapter 2: Section 2.3). These threat models should capture the strongest plausible $\mathcal{A}dv$ in a given deployment setting; otherwise, defenses may be evaluated only against weak attacks, leading to a false sense of robustness (e.g., [126, 127]).
2. **Identifying Requirements for Defense Design:** Defenses should be designed and evaluated according to precise requirements tailored to the application (e.g., efficiency and utility preservation). Here, robustness should be included as a fundamental requirement that ensures protection against the strongest attacks.

Trustworthy ML as a Sociotechnical Problem: ML models pose significant societal and safety risks. They are prone to hallucinations that users may accept as factual [578, 579, 580, 581, 582, 583, 584, 585], enabling misinformation and amplifying divisive narratives [586, 587, 588]. Malicious uses, such as propaganda [589] or sophisticated cyberattacks [590, 591, 592, 593], further threaten safety. Harmful or inappropriate outputs [594, 595, 101], and unregulated applications in therapy or mental health support [596, 597, 598, 599], pose direct risks to users. Collectively, these underscore the substantial societal and safety risks of ML models, that go beyond technical challenges. Thus, ensuring the trustworthy deployment of ML systems is a sociotechnical problem requiring interdisciplinary collaboration (e.g., social sciences). Thus, it is important to understand how ML systems may deviate from or reshape societal norms in both beneficial and harmful ways. For instance:

- Alignment with human values is particularly challenging because they cannot be precisely formalized, leading to issues such as reward hacking [600], and because they are dynamic and often conflicting across populations. Recent efforts, such as constitutions for LLMs (e.g., the Claude Constitution [601], provide a starting point that could be extended to more comprehensively incorporate societal values into LLMs. Incorporating insights from the social sciences may help design objectives for aligning ML models with diverse human values.
- The design of accountability mechanisms could use input from multiple stakeholders, including developers and end users, to ensure system behavior aligns with expectations. For regulatory compliance, collaboration with legal scholars is also necessary to interpret existing regulations and assess how technical mechanisms satisfy them, potentially requiring iterative refinement of their designs based on legal feedback.

References

[1] **Vasisht Duddu**, Sebastian Szyller, and N Asokan. SoK: Unintended Interactions among Machine Learning Defenses and Risks. In ***IEEE Symposium on Security and Privacy (S&P)*** ***[Distinguished Paper Award]***, 2024.

[2] **Vasisht Duddu**, Rui Zhang, and N Asokan. Combining Machine Learning Defenses without Conflicts. ***Transactions on Machine Learning Research (TMLR)***, 2025.

[3] **Vasisht Duddu**, Lipeng He, Asim Waheed, and N Asokan. SoK: Colluding Adversaries in Machine Learning Pipelines. In ***USENIX Security Symposium (USENIX SEC)***, 2026.

[4] **Vasisht Duddu**, Anudeep Das, Nora Khayata, Hossein Yalame, Thomas Schneider, and N Asokan. Attesting Distributional Properties of Training Data for Machine Learning. In ***European Symposium on Research in Computer Security (ESORICS)***, 2024.

[5] **Vasisht Duddu**, Oskari Jarvinen, Lachlan Gunn, and N Asokan. Laminator: Verifiable ML Property Cards using Hardware-assisted Attestations. In ***ACM Conference on Data and Application Security and Privacy (CODASPY)***, 2025.

[6] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. SoK: Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414, 2018.

[7] Fnu Suya, Anshuman Suri, Tingwei Zhang, Jingtao Hong, Yuan Tian, and David Evans. SoK: Pitfalls in Evaluating Black-Box Attacks. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 387–407, 2024.

[8] Lucas Fenaux and Florian Kerschbaum. SoK: Analyzing Adversarial Examples: A Framework to Study Adversary Knowledge. *arXiv preprint*, 2024.

[9] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A. Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning. *ACM Comput. Surv.*, 55(13s), July 2023.

[10] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences. *ACM Comput. Surv.*, 55(14s), July 2023.

[11] Maria Rigaki and Sebastian Garcia. A Survey of Privacy Attacks in Machine Learning. *ACM Comput. Surv.*, 56(4), 2023.

[12] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. In *Privacy Enhancing Technologies Symposium*, 2022.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[14] Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. In *European Conference on Computer Vision (ECCV)*, pages 74–92, 2022.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6840–6851, 2020.

[17] Irina Jurenka, Markus Kunesch, Kevin R McKee, Daniel Gillick, Shaojian Zhu, Sara Wiltberger, Shubham Milind Phal, Katherine Hermann, Daniel Kasenberg, Avishkar Bhoopchand, et al. Towards responsible development of generative AI for education: An evaluation-driven approach. *arXiv preprint*, 2024.

[18] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

[19] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[20] Google Cloud. AI in Finance: Applications, Examples & Benefits. https://cloud.google.com/discover/finance-ai, 2026. Accessed: February 7, 2026.

[21] McKinsey & Company. How finance teams are putting AI to work today. https://www.mckinsey.com/capabilities/strategy-and-corporate-finance/our-insights/how-finance-teams-are-putting-ai-to-work-today, nov 2025. Accessed: February 7, 2026.

[22] IBM. Artificial Intelligence in Finance. https://www.ibm.com/think/topics/artificial-intelligence-finance, 2026. Accessed: February 7, 2026.

[23] Parul Awasthy, Aashka Trivedi, Yulong Li, Mihaela Bornea, David Cox, Abraham Daniels, Martin Franz, Gabe Goodhart, Bhavani Iyer, Vishwajeet Kumar, et al. Granite Embedding Models. *arXiv preprint*, 2025.

[24] Bing Zhang, Mikio Takeuchi, Ryo Kawahara, Shubhi Asthana, Md. Maruf Hossain, Guang-Jie Ren, Kate Soule, Yifan Mai, Yada Zhu, Yi Yang, Mohammad Kachuee, and Xue-Yong Fu. Evaluating Large Language Models with Enterprise Benchmarks. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, pages 485–505, 2025.

[25] OpenAI. ChatGPT. https://chatgpt.com/, 2026. Accessed: 2026-02-06.

[26] Google. Gemini. https://gemini.google.com/, 2026. Accessed: 2026-02-06.

[27] Perplexity AI, Inc. Perplexity AI. https://www.perplexity.ai/, 2026. Accessed: 2026-02-06.

[28] Consensus Inc. Consensus: AI for Research. https://consensus.app/, 2026. Accessed: 2026-02-06.

[29] Perplexity AI Team. Introducing Comet: Browse at the speed of thought. https://www.perplexity.ai/hub/blog/introducing-comet, January 2026. Accessed: 2026-02-06.

[30] OpenAI. Introducing ChatGPT Atlas. https://openai.com/index/introducing-chatgpt-atlas/, October 2025. Accessed: 2026-02-06.

[31] Yixiao Song, Katherine Thai, Chau Minh Pham, Yapei Chang, Mazin Nadaf, and Mohit Iyyer. BEARCUBS: A benchmark for computer-using web agents. In *Conference on Language Modeling (COLM)*, 2025.

[32] Tara Bogavelli, Roshnee Sharma, and Hari Subramani. AgentArch: a comprehensive benchmark to evaluate agent architectures in enterprise. *arXiv preprint*, 2025.

[33] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 6129–6139, 2025.

[34] OpenAI. Openai for healthcare. https://openai.com/index/openai-for-healthcare/, 2026. Accessed: February 6, 2026.

[35] U.A. Lyakhova and P.A. Lyakhov. Systematic review of approaches to detection and classification of skin cancer using artificial intelligence: Development and prospects. *Computers in Biology and Medicine*, 178:108742, 2024.

[36] Hariram Sankar, Ragavi Alagarsamy, Babu Lal, Shailendra Singh Rana, Ajoy Roychoudhury, Amit Agrawal, and Syrpailyne Wankhar. Role of artificial intelligence in treatment planning and outcome prediction of jaw corrective surgeries by using 3-D imaging: a systematic review. *Oral Surgery, Oral Medicine, Oral Pathology and Oral Radiology*, 139(3):299–310, 2025.

[37] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. AI is sending people to jail—and getting it wrong. https://www.technologyreview.com/2019/01/21/137783/algorithms-criminal-justice-ai/, January 2019. Accessed: February 6, 2026.

[38] Thomson Reuters. Humanizing Justice: The transformational impact of AI in courts. https://www.thomsonreuters.com/en-us/posts/ai-in-courts/humanizing-justice/, Jan. 2026.

[39] Melissa Hamilton. A black box AI system has been influencing criminal justice decisions for over two decades—it's time to open it up.

[40] Tomas Chamorro-Premuzic. AI Has Made Hiring Worse—But It Can Still Help. *Harvard Business Review*, jan 2026. Accessed: February 6, 2026.

[41] The Globe and Mail. AI Becoming Unavoidable for Recruiters and Job Seekers. *The Globe and Mail*, aug 2025. Accessed: February 6, 2026.

[42] Bytebot. Bytebot - Desktop agents that use computers like a human. https://www.bytebot.ai/, 2025.

[43] Warmwind. warmwind: Autonomous cloud employees. https://warmwind.space/, 2025.

[44] Tesla. Ai & robotics. https://www.tesla.com/en_ca/AI, 2026. Accessed: February 6, 2026.

[45] Waymo. Behind the Innovation: AI & ML at Waymo. https://waymo.com/blog/2024/10/ai-and-ml-at-waymo, oct 2024. Accessed: February 6, 2026.

[46] Alexander Ratner, Dan Alistarh, Gustavo Alonso, David G Andersen, Peter Bailis, Sarah Bird, Nicholas Carlini, Bryan Catanzaro, Jennifer Chayes, Eric Chung, et al. MLsys: The new frontier of machine learning systems. *arXiv preprint*, 2019.

[47] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484, 2023.

[48] Chengliang Chai, Jiayi Wang, Yuyu Luo, Zeping Niu, and Guoliang Li. Data Management for Machine Learning: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 35(5):4646–4667, 2023.

[49] Rui Wen, Yiyong Liu, Michael Backes, Yang Zhang, and Giancarlo Pellegrino. SoK: Data Reconstruction Attacks Against Machine Learning Models: Definition, Metrics, and Benchmark. In *USENIX Security Symposium (USENIX SEC)*, pages 5601–5620, 2025.

[50] Maximilian Noppel and Christian Wressnegger. SoK: Explainable Machine Learning in Adversarial Environments. In *IEEE Symposium on Security and Privacy (S&P)*, pages 2441–2459, 2024.

[51] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership Inference Attacks on Machine Learning: A Survey. *ACM Comput. Surv.*, jan 2022.

[52] Linkang Du, Xuanru Zhou, Min Chen, Chusong Zhang, Zhou Su, Peng Cheng, Jiming Chen, and Zhikun Zhang. Sok: Dataset copyright auditing in machine learning systems. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1–19, 2025.

[53] Linyi Li, Tao Xie, and Bo Li. SoK: Certified Robustness for Deep Neural Networks . In *IEEE Symposium on Security and Privacy (S&P)*, pages 1289–1310, 2023.

[54] Gabriel Resende Machado, Eugênio Silva, and Ronaldo Ribeiro Goldschmidt. Adversarial Machine Learning in Image Classification: A Survey Toward the Defender's Perspective. *ACM Comput. Surv.*, 55(1), nov 2021.

[55] Simon Caton and Christian Haas. Fairness in Machine Learning: A Survey. *ACM Comput. Surv.*, 56(7), April 2024.

[56] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *ACM Comput. Surv.*, 55(8), December 2022.

[57] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.*, 51(5), aug 2018.

[58] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 241–257, 2019.

[59] Jamie Hayes. Trade-offs between membership privacy & adversarially robust learning. *arXiv preprint*, 2020.

[60] Xinsong Ma, Zekai Wang, and Weiwei Liu. On the Tradeoff Between Robustness and Fairness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[61] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Robustness May Be at Odds with Fairness: An Empirical Study on Class-wise Accuracy. In *NeurIPS Workshop on Pre-registration in Machine Learning*, pages 325–342, 2021.

[62] Kacem Khaled, Gabriela Nicolescu, and Felipe Gohring De Magalhães. Careful what you wish for: on the extraction of adversarially trained models. In *Annual International Conference on Privacy, Security & Trust (PST)*, pages 1–10, 2022.

[63] Asim Waheed, Vasisht Duddu, Rui Zhang, and Sebastian Szyller. Amulet: a Python Library for Assessing Interactions Among ML Defenses and Risks. *arXiv preprint*, 2025.

[64] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[65] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model Cards for Model Reporting. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 220–229, 2019.

[66] Daniel Huynh and Jacob Hardouin. PoisonGPT: How We Hid a Lobotomized LLM on Hugging Face to Spread Fake News, dec 2023.

[67] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. The Untapped Potential of Trusted Execution Environments on Mobile Devices. *IEEE Security & Privacy*, 12(4):29–37, 2014.

[68] N. Asokan, Jan-Erik Ekberg, Kari Kostiainen, Anand Rajan, Carlos Rozas, Ahmad-Reza Sadeghi, Steffen Schulz, and Christian Wachsmann. Mobile Trusted Computing. *Proceedings of the IEEE*, 102(8):1189–1206, 2014.

[69] Joanna Young and Sam Lugani. How Confidential Accelerators can boost AI workload security. `https://cloud.google.com/blog/products/identity-security/how-confidential-accelerators-can-boost-ai-workload-security`, 06 2024. Google Cloud Blog.

[70] Gerrit J.J. Van den Burg and Chris Williams. On Memorization in Probabilistic Deep Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[71] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train Faster, Generalize Better: Stability of Stochastic Gradient Descent. In *International Conference on Machine Learning (ICML)*, page 1225–1234, 2016.

[72] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium (USENIX SEC)*, pages 267–284, 2019.

[73] Stefan Arnold, Eric Wallace, Yangsibo Huang, Tiago Pimentel, Pratyush Maini, Verna Dankers, Johnny Wei, and Pietro Lesci. Memorization in Language Models through the Lens of Intrinsic Dimension. In *Workshop on Large Language Model Memorization (L2M2)*, pages 23–28, 2025.

[74] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning? In *Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 123–132, 2021.

[75] Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A Closer Look at Memorization in Deep Networks. In *International Conference on Machine Learning (ICML)*, page 233–242, 2017.

[76] Nicholas Carlini, Matthew Jagielski, Chiyuan Zhang, Nicolas Papernot, Andreas Terzis, and Florian Tramer. The Privacy Onion Effect: Memorization is Relative. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[77] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 954–959, 2020.

[78] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[79] Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[80] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying Memorization Across Neural Language Models. In *International Conference on Learning Representations (ICLR)*, 2023.

[81] William H Rogers and Terry J Wagner. A finite sample distribution-free performance bound for local discrimination rules. *The Annals of Statistics*, 6(3):506–514, 1978.

[82] Luc Devroye and Terry Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

[83] Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1999.

[84] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research (JMLR)*, 2(Mar):499–526, 2002.

[85] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:2881–2891, 2020.

[86] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *USENIX Security Symposium (USENIX SEC)*, pages 2633–2650, 2021.

[87] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *USENIX Security Symposium (USENIX SEC)*, pages 5253–5270, 2023.

[88] Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning (ICML)*, 2022.

[89] Alexander Xiong, Xuandong Zhao, Aneesh Pappu, and Dawn Song. The landscape of memorization in LLMs: Mechanisms, measurement, and mitigation. *arXiv preprint*, 2025.

[90] Shruti Tople, Amit Sharma, and Aditya V. Nori. Alleviating Privacy Attacks via Causal Learning. In *International Conference on Machine Learning (ICML)*, 2020.

[91] Divyat Mahajan, Shruti Tople, and Amit Sharma. The Connection between Out-of-Distribution Generalization and Privacy of ML Models. In *NeurIPS Workshop on Privacy Preserving Machine Learning*, 2020.

[92] Jian Liu, Rui Zhang, Sebastian Szyller, Kui Ren, and N. Asokan. False Claims against Model Ownership Resolution. In *USENIX Security Symposium (USENIX SEC)*, pages 6885–6902, 2024.

[93] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security Symposium (USENIX SEC)*, pages 1505–1521, 2021.

[94] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. Practical Blind Membership Inference Attack via Differential Comparisons. In *Network and Distributed System Security Symposium (NDSS)*, 2021.

[95] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. Sok: The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems. In *IEEE Symposium on Security and Privacy (S&P)*, pages 730–747, 2021.

[96] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[97] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management.

[98] Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. Membership privacy: a unifying framework for privacy definitions. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 889–900, 2013.

[99] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[100] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint*, 2014.

[101] Minh Pham, Kelly O. Marshall, Niv Cohen, Govind Mittal, and Chinmay Hegde. Circumventing Concept Erasure Methods For Text-To-Image Generative Models. In *International Conference on Learning Representations (ICLR)*, 2024.

[102] Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint*, 2024.

[103] Gorka Abad, Jing Xu, Stefanos Koffas, Behrad Tajalli, Stjepan Picek, and Mauro Conti. Sok: A systematic evaluation of backdoor trigger characteristics in image classification. *arXiv preprint*, 2023.

[104] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems (TNNLS)*, 35(1):5–22, 2022.

[105] Shuai Zhao, Xiaobao Wu, Cong-Duy T Nguyen, Yanhao Jia, Meihuizi Jia, Feng Yichao, and Anh Tuan Luu. Unlearning Backdoor Attacks for LLMs with Weak-to-Strong Knowledge Distillation. In *ACL*, pages 4937–4952, 2025.

[106] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4954–4963, 2019.

[107] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *International Conference on Learning Representations (ICLR)*, 2020.

[108] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. PRADA: protecting against DNN model stealing attacks. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527, 2019.

[109] Nicholas Carlini, Daniel Paleka, Krishnamurthy (Dj) Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing part of a production language model. In *International Conference on Machine Learning (ICML)*, 2024.

[110] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning (ICML)*, pages 8326–8335, 2020.

[111] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable Examples: Making Personal Data Unexploitable. In *International Conference on Learning Representations (ICLR)*, 2021.

[112] Emily Wenger, Shawn Shan, Haitao Zheng, and Ben Y Zhao. Sok: Anti-facial recognition technology. In *IEEE Symposium on Security and Privacy (S&P)*, pages 864–881, 2023.

[113] Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do Membership Inference Attacks Work on Large Language Models? In *Conference on Language Modeling (COLM)*, 2024.

[114] Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *International Conference on Machine Learning (ICML)*, pages 8717–8730, 2023.

[115] Bargav Jayaraman and David Evans. Are attribute inference attacks just imputation? In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1569–1582, 2022.

[116] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 346–363, 2023.

[117] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural Network Inversion in Adversarial Setting via Background Knowledge Alignment. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 225–240, 2019.

[118] Vincenzo Carletti, Pasquale Foggia, Carlo Mazzocca, Giuseppe Parrella, and Mario Vento. SoK: Gradient inversion attacks in federated learning. In *USENIX Security Symposium (USENIX SEC)*, 2025.

[119] Michael Aerni, Javier Rando, Edoardo Debenedetti, Nicholas Carlini, Daphne Ippolito, and Florian Tramèr. Measuring Non-Adversarial Reproduction of Training Data in Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2025.

[120] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 691–706, 2019.

[121] Anshuman Suri, Yifu Lu, Yanjin Chen, and David Evans. Dissecting distribution inference. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 150–164, 2023.

[122] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 619–633, 2018.

[123] Lijin Wang, Jingjing Wang, Jie Wan, Lin Long, Ziqi Yang, and Zhan Qin. Property Existence Inference against Generative Models. In *USENIX Security Symposium (USENIX SEC)*, pages 2423–2440, 2024.

[124] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.*, 54(6), July 2021.

[125] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 739–753, 2019.

[126] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1633–1645, 2020.

[127] Ryan Feng, Ashish Hooda, Neal Mangaokar, Kassem Fawaz, Somesh Jha, and Atul Prakash. Stateful defenses for machine learning models are not yet secure against black-box attacks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 786–800, 2023.

[128] Nils Lukas, Abdulrahman Diaa, Lucas Fenaux, and Florian Kerschbaum. Leveraging Optimization for Adaptive Attacks on Image Watermarks. In *International Conference on Learning Representations (ICLR)*, 2024.

[129] Gilad Asharov et al. Static vs. Adaptive Security in Perfect MPC: A Separation and the Adaptive Security of BGW. In *Cryptology ePrint Archive, Paper 2022/758*, 2022.

[130] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Annual International Cryptology Conference (CRYPTO)*, pages 1–12, 1998.

[131] S. Yun, D. Han, S. Chun, S. Oh, Y. Yoo, and J. Choe. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031, 2019.

[132] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

[133] Terrance DeVries and Graham W Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv preprint*, 2017.

[134] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 29935–29948, 2021.

[135] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. In *International Conference on Machine Learning (ICML)*, pages 7472–7482, 2019.

[136] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *International Conference on Machine Learning (ICML)*, pages 1310–1320, 2019.

[137] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (S&P)*, pages 656–672, 2019.

[138] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

[139] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion Models for Adversarial Purification. In *International Conference on Machine Learning (ICML)*, 2022.

[140] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[141] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[142] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. In *International Conference on Learning Representations (ICLR)*, 2018.

[143] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression. *arXiv preprint*, 2017.

[144] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint*, 2017.

[145] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8239–8247, 2021.

[146] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proc. VLDB Endow.*, 12(11):1610–1623, July 2019.

[147] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference (ACSAC)*, pages 897–912, 2020.

[148] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. Casting out Demons: Sanitizing Training Data for Anomaly Sensors. In *IEEE Symposium on Security and Privacy (S&P)*, pages 81–95, 2008.

[149] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv preprint*, 2018.

[150] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 8011–8021, 2018.

[151] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, 2010.

[152] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint*, 2018.

[153] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L. Li. Learning from Noisy Labels with Distillation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1928–1936, 2017.

[154] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A Robust Meta-Algorithm for Stochastic Optimization. In *International Conference on Machine Learning (ICML)*, pages 1596–1606, 2019.

[155] Mingli Zhu, Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. Neural Polarizer: A Lightweight and Effective Backdoor Defense via Purifying Poisoned Features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[156] Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. L_dmi: a novel information-theoretic loss function for training deep nets robust to label noise. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[157] Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *International Conference on Machine Learning (ICML)*, pages 6226–6236, 2020.

[158] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1944–1952, 2017.

[159] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294, 2018.

[160] Dongxian Wu and Yisen Wang. Adversarial Neuron Pruning Purifies Backdoored Deep Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[161] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Pre-activation Distributions Expose Backdoor Neurons. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[162] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision (ECCV)*, pages 175–191, 2022.

[163] Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. Reconstructive neuron pruning for backdoor defense. In *International Conference on Machine Learning (ICML)*, pages 19837–19854, 2023.

[164] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX SEC)*, pages 1615–1631, 2018.

[165] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, page 159–172, 2018.

[166] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium (USENIX SEC)*, pages 1937–1954, 2021.

[167] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding Watermarks into Deep Neural Networks. In *ACM on International Conference on Multimedia Retrieval (ICMR)*, page 269–277, 2017.

[168] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning (ICML)*, pages 1450–1465, 2022.

[169] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. DAWN: Dynamic Adversarial Watermarking of Neural Networks. In *ACM International Conference on Multimedia (MM)*, page 4417–4425, 2021.

[170] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting Intellectual Property of Deep Neural Networks via Fingerprinting the Classification Boundary. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, page 14–25, 2021.

[171] Zirui Peng, Shaofeng Li, Guoxing Chen, Cheng Zhang, Haojin Zhu, and Minhui Xue. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13430–13439, 2022.

[172] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2021.

[173] Yue Zheng, Si Wang, and Chip-Hong Chang. A DNN Fingerprint for Non-Repudiable Model Ownership Identification and Piracy Detection. *IEEE Transactions on Information Forensics and Security (TIFS)*, 17:2977–2989, 2022.

[174] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset Inference: Ownership Resolution in Machine Learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[175] Buse Gul Atli Tekgul and N. Asokan. On the Effectiveness of Dataset Watermarking. In *ACM on International Workshop on Security and Privacy Analytics (IWSPA)*, page 93–99, 2022.

[176] Gaoyang Liu, Tianlong Xu, Xiaoqiang Ma, and Chen Wang. Your Model Trains on My Data? Protecting Intellectual Property of Training Data via Membership Fingerprint Authentication. *IEEE Transactions on Information Forensics and Security (TIFS)*, 17:1024–1037, 2022.

[177] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318, 2016.

[178] Yuzheng Hu, Fan Wu, Qinbin Li, Yunhui Long, Gonzalo Munilla Garrido, Chang Ge, Bolin Ding, David Forsyth, Bo Li, and Dawn Song. SoK: Privacy-Preserving Data Synthesis . In *IEEE Symposium on Security and Privacy (S&P)*, pages 4696–4713, 2024.

[179] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially Private Generative Adversarial Network. *arXiv preprint*, abs/1802.06739, 2018.

[180] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: Differentially Private Synthetic Data and Label Generation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[181] Tianhang Zheng and Baochun Li. Differentially Private Dataset Condensation. In *Network and Distributed System Security Symposium (NDSS)*, 2023.

[182] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *International Conference on Learning Representations (ICLR)*, 2017.

[183] Bargav Jayaraman and David Evans. Evaluating Differentially Private Machine Learning in Practice. In *USENIX Security Symposium (USENIX SEC)*, pages 1895–1912, 2019.

[184] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research (JMLR)*, 12(3), 2011.

[185] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3855–3859, 2021.

[186] Han Qiu, Yi Zeng, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. DeepSweep: An Evaluation Framework for Mitigating DNN Backdoor Attacks using Data Augmentation. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, page 363–377, 2021.

[187] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label sanitization against label flipping poisoning attacks. In *ECML PKDD Workshops*, pages 5–15, 2019.

[188] Faisal Kamiran and Toon Calders. Data Pre-Processing Techniques for Classification without Discrimination. *Knowledge and Information Systems*, 33, 10 2011.

[189] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized Pre-Processing for Discrimination Prevention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[190] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning Fair Representations. In *International Conference on Machine Learning (ICML)*, pages 325–333, 2013.

[191] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[192] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 319–328, 2019.

[193] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. In *International Conference on Machine Learning (ICML)*, pages 2564–2572, 2018.

[194] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. An empirical study of rich subgroup fairness for machine learning. In *ACM Conference on Fairness, Accountability, and Transparency (FaccT)*, pages 100–109, 2019.

[195] Alekh Agarwal, Miroslav Dudík, and Zhiwei Steven Wu. Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning (ICML)*, pages 120–129, 2019.

[196] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning (ICML)*, pages 60–69, 2018.

[197] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pages 335–340, 2018.

[198] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-Aware Classifier with Prejudice Remover Regularizer. In *Machine Learning and Knowledge Discovery in Databases*, pages 35–50, 2012.

[199] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[200] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[201] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision Theory for Discrimination-Aware Classification. In *IEEE International Conference on Data Mining (ICDM)*, pages 924–929, 2012.

[202] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2221–2231, 2019.

[203] Aya Abdelsalam Ismail, Hector Corrada Bravo, and Soheil Feizi. Improving Deep Learning Interpretability by Saliency Guided Training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[204] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv preprint*, 2017.

[205] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning (ICML)*, page 3319–3328, 2017.

[206] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning (ICML)*, pages 1885–1894, 2017.

[207] Sandra Wachter, Brent Daniel Mittelstadt, and Chris Russell. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *Cybersecurity*, 2017.

[208] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[209] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning (ICML)*, pages 2668–2677, 2018.

[210] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness Constraints: A Flexible Approach for Fair Classification . *Journal of Machine Learning Research*, 20(75):1–42, 2019.

[211] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating Unwanted Biases with Adversarial Learning. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 335–340, 2018.

[212] Gilles Louppe, Michael Kagan, and Kyle Cranmer. Learning to pivot with adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[213] Carlos Pinzón, Catuscia Palamidessi, Pablo Piantanida, and Frank Valencia. On the incompatibility of accuracy and equal opportunity. *Mach. Learn.*, 113(5):2405–2434, may 2023.

[214] Tiago Salvador, Stephanie Cairns, Vikram Voleti, Noah Marshall, and Adam M Oberman. FairCal: Fairness Calibration for Face Verification. In *International Conference on Learning Representations (ICLR)*, 2022.

[215] Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pages 623–631, 2017.

[216] Ursula Hebert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (Computationally-Identifiable) Masses. In *International Conference on Machine Learning (ICML)*, pages 1939–1948, 2018.

[217] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. In *Workshop on Visualization for Deep Learning, ICML*, 2017.

[218] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19920–19930, 2020.

[219] Alex Gittens, Bülent Yener, and Moti Yung. An Adversarial Perspective on Accuracy, Robustness, Fairness, and Privacy: Multilateral-Tradeoffs in Trustworthy ML. *IEEE Access*, 10:120850–120865, 2022.

[220] M. Strobel and R. Shokri. Data Privacy and Trustworthy Machine Learning. In *IEEE Security & Privacy Magazine*, pages 2–7, 2022.

[221] Huiqiang Chen, Tianqing Zhu, Tao Zhang, Wanlei Zhou, and Philip S. Yu. Privacy and Fairness in Federated Learning: On the Perspective of Tradeoff. *ACM Comput. Surv.*, 56(2), sep 2023.

[222] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 1992.

[223] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking Bias-Variance Trade-off for Generalization of Neural Networks. In *International Conference on Machine Learning (ICML)*, 2020.

[224] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences (PNAS)*, 116(32):15849–15854, 2019.

[225] Yehuda Dar, Vidya Muthukumar, and Richard G Baraniuk. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. *arXiv preprint*, 2021.

[226] Xinli Yue, Ningping Mou, Qian Wang, and Lingchen Zhao. Revisiting Adversarial Training Under Long-Tailed Distributions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24492–24501, 2024.

[227] Tong Wu, Ziwei Liu, Qingqiu Huang, Yu Wang, and Dahua Lin. Adversarial robustness under long-tailed distribution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8659–8668, 2021.

[228] Shelvia Wongso, Rohan Ghosh, and Mehul Motani. Using Sliced Mutual Information to Study Memorization and Generalization in Deep Neural Networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 11608–11629, 2023.

[229] V. Hartmann, L. Meynent, M. Peyrard, D. Dimitriadis, S. Tople, and R. West. Distribution Inference Risks: Identifying and Mitigating Sources of Leakage. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 136–149, 2023.

[230] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching. In *International Conference on Machine Learning (ICML)*, pages 7313–7324, 2021.

[231] Ferdinando Fioretto, Cuong Tran, Pascal Van Hentenryck, and Keyu Zhu. Differential Privacy and Fairness in Decisions and Learning Tasks: A Survey. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5470–5477, 2022. Survey Track.

[232] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 2779–2792, 2022.

[233] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards Efficient Data Valuation Based on the Shapley Value. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1167–1176, 2019.

[234] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating Training Data Makes Language Models Better. In *Association for Computational Linguistics (ACL)*, pages 8424–8445, 2022.

[235] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[236] Sebastian Szyller and N Asokan. Conflicting interactions among protection mechanisms for machine learning models. *AAAI Conference on Artificial Intelligence (AAAI)*, 37(12):15179–15187, 2023.

[237] Yuzheng Hu, Fan Wu, Hongyang Zhang, and Han Zhao. Understanding the Impact of Adversarial Robustness on Accuracy Disparity. In *International Conference on Machine Learning (ICML)*, 2023.

[238] Yuancheng Xu, Yanchao Sun, Micah Goldblum, Tom Goldstein, and Furong Huang. Exploring and Exploiting Decision Boundary Dynamics for Adversarial Robustness. In *International Conference on Learning Representations (ICLR)*, 2023.

[239] Haotao Wang, Aston Zhang, Yi Zhu, Shuai Zheng, Mu Li, Alex J Smola, and Zhangyang Wang. Partial and asymmetric contrastive learning for out-of-distribution detection in long-tailed recognition. In *International Conference on Machine Learning (ICML)*, pages 23446–23458, 2022.

[240] Vasisht Duddu, Sebastian Szyller, and N Asokan. SHAPr: An Efficient and Versatile Membership Privacy Risk Metric for Machine Learning. *arXiv preprint*, 2021.

[241] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX SEC)*, pages 1615–1631, 2018.

[242] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *IEEE Symposium on Security and Privacy (S&P)*, pages 787–804, 2022.

[243] Sofiane Lounici, Melek Önen, Orhan Ermis, and Slim Trabelsi. BlindSpot: Watermarking Through Fairness. In *ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec)*, page 39–50, 2022.

[244] Asim Waheed, **Vasisht Duddu**, and N Asokan. GrOVe: Ownership Verification of Graph Neural Networks using Embeddings. In ***IEEE Symposium on Security and Privacy (S&P)***, 2024.

[245] Yilin Wang and Farzan Farnia. On the Role of Generalization in Transferability of Adversarial Examples. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 2259–2270, 2023.

[246] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *USENIX Security Symposium (USENIX SEC)*, 2019.

[247] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership Inference Attack against Differentially Private Deep Learning Model. *Trans. Data Priv.*, 11(1):61–79, 2018.

[248] Nurislam Tursynbek, Aleksandr Petiushko, and Ivan Oseledets. Robustness threats of differential privacy. *NeurIPS'20 Privacy-Preserving Machine Learning Workshop*, 2020.

[249] Franziska Boenisch, Philip Sperl, and Konstantin Böttinger. Gradient masking and the underestimated robustness threats of differential privacy in deep learning. *arXiv preprint*, 2021.

[250] Vinith M. Suriyakumar, Nicolas Papernot, Anna Goldenberg, and Marzyeh Ghassemi. Chasing Your Long Tails: Differentially Private Prediction in Health Care Settings. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 723–734, 2021.

[251] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential Privacy Has Disparate Impact on Model Accuracy. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15479–15488, 2019.

[252] Yinchen Shen, Zhiguo Wang, Ruoyu Sun, and Xiaojing Shen. Towards Understanding the Impact of Model Size on Differential Private Classification. *arXiv preprint*, 2021.

[253] Runtian Zhai, Chen Dan, Zico Kolter, and Pradeep Ravikumar. Understanding why generalized reweighting does not improve over ERM. In *International Conference on Learning Representations (ICLR)*, 2023.

[254] Akshaj Kumar Veldanda, Ivan Brugere, Jiahao Chen, Sanghamitra Dutta, Alan Mishler, and Siddharth Garg. Fairness via In-Processing in the Over-parameterized Regime: A Cautionary Tale with MinDiff Loss. *Transactions on Machine Learning Research (TMLR)*, 2023.

[255] Kit T Rodolfa, Hemank Lamba, and Rayid Ghani. Empirical observation of negligible fairness–accuracy trade-offs in machine learning for public policy. *Nature Machine Intelligence*, 3(10):896–904, 2021.

[256] Jan Aalmoes, Vasisht Duddu, and Antoine Boutet. On the alignment of group fairness with attribute privacy. In *International conference on web information systems engineering (WISE)*, pages 333–348, 2025.

[257] Razieh Nabi and Ilya Shpitser. Fair inference on outcomes. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[258] Hongyan Chang and Reza Shokri. On the privacy risks of algorithmic fairness. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 292–303, 2021.

[259] Cuong Tran, Keyu Zhu, Pascal Van Hentenryck, and Ferdinando Fioretto. On the effects of fairness to adversarial vulnerability. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[260] Reza Shokri, Martin Strobel, and Yair Zick. On the Privacy Risks of Model Explanations. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 231–241, 2021.

[261] Vasisht Duddu and Antoine Boutet. Inferring Sensitive Attributes from Model Explanations. In *ACM International Conference on Information & Knowledge Management (CIKM)*, page 416–425, 2022.

[262] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 241–257, 2019.

[263] Sainyam Galhotra, Karthikeyan Shanmugam, Prasanna Sattigeri, and Kush R. Varshney. Causal Feature Selection for Algorithmic Fairness. In *International Conference on Management of Data (SIGMOD)*, page 276–285, 2022.

[264] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[265] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, pages 268–282, 2018.

[266] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 3–18, 2017.

[267] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1897–1914, 2022.

[268] Yuxin Wen, Jonas Geiping, Liam H. Fowl, Micah Goldblum, and Tom Goldstein. Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification. In *International Conference on Machine Learning (ICML)*, 2022.

[269] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 1322–1333, 2015.

[270] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[271] Ngoc-Bao Nguyen, Keshigeyan Chandrasegaran, Milad Abdollahzadeh, and Ngai-Man Cheung. Re-Thinking Model Inversion Attacks Against Deep Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16384–16393, 2023.

[272] Ziqi Yang, Lijin Wang, Da Yang, Jie Wan, Ziming Zhao, Ee-Chien Chang, Fan Zhang, and Kui Ren. Purifier: defending data inference attacks via transforming confidence scores. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 10871–10879, 2023.

[273] Congzheng Song and Vitaly Shmatikov. Overlearning Reveals Sensitive Attributes. In *International Conference on Learning Representations (ICLR)*, 2020.

[274] H. Chaudhari, J. Abascal, A. Oprea, M. Jagielski, F. Tramer, and J. Ullman. SNAP: Efficient Extraction of Private Properties with Poisoning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 400–417, 2023.

[275] Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. Property Inference from Poisoning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1120–1137, 2022.

[276] Hamid Karimi and Jiliang Tang. Decision Boundary of Deep Neural Networks: Challenges and Opportunities. In *International Conference on Web Search and Data Mining (WSDM)*, page 919–920, 2020.

[277] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1277–1294, 2020.

[278] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. {ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models. In *USENIX Security Symposium (USENIX SEC)*, pages 4525–4542, 2022.

[279] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine Learning Models That Remember Too Much. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 587–601, 2017.

[280] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chee-Kong Lee, and Enhong Chen. Model inversion attacks against graph neural networks. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

[281] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis. In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[282] Mohammad Yaghini, Bogdan Kulynych, and Carmela Troncoso. Disparate Vulnerability: on the Unfairness of Privacy Attacks Against Machine Learning. In *Privacy Enhancing Technologies Symposium*, 2022.

[283] Rui Wen, Zhengyu Zhao, Zhuoran Liu, Michael Backes, Tianhao Wang, and Yang Zhang. Is Adversarial Training Really a Silver Bullet for Mitigating Data Poisoning? In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[284] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Better Safe Than Sorry: Preventing Delusive Adversaries with Adversarial Training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[285] Jingyang Zhang, Yiran Chen, and Hai Li. Privacy Leakage of Adversarial Training Models in Federated Learning Systems. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 108–114, 2022.

[286] Felipe A Mejia, Paul Gamble, Zigfried Hampel-Arias, Michael Lomnitz, Nina Lopatina, Lucas Tindall, and Maria Alejandra Barrios. Robust or private? adversarial training makes models more vulnerable to privacy attacks. *arXiv preprint*, 2019.

[287] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[288] Shubhranshu Shekhar, Neil Shah, and Leman Akoglu. FairOD: Fairness-Aware Outlier Detection. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 210–220, 2021.

[289] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 6106–6116, 2018.

[290] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *Asia Conference on Computer and Communications Security (ASIACCS)*, page 159–172, 2018.

[291] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13238–13250, 2022.

[292] Yufei Chen, Chao Shen, Yun Shen, Cong Wang, and Yang Zhang. Amplifying Membership Exposure via Data Poisoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[293] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium (USENIX SEC)*, page 601–618, 2016.

[294] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2020.

[295] Shubhranshu Shekhar, Neil Shah, and Leman Akoglu. Fairod: Fairness-aware outlier detection. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pages 210–220, 2021.

[296] Hongyan Chang, Ta Duy Nguyen, Sasi Kumar Murakonda, Ehsan Kazemi, and Reza Shokri. On adversarial bias and the robustness of fair machine learning. *arXiv preprint*, 2020.

[297] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. Subpopulation data poisoning attacks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 3104–3122, 2021.

[298] Zhiqi Bu and Yuan Zhang. Differentially Private Optimizers Can Learn Adversarially Robust Models. *Transactions on Machine Learning Research (TMLR)*, 2023.

[299] Matthew Jagielski and Alina Oprea. Does Differential Privacy Defeat Data Poisoning? *Workshop on Distributed and Private Machine Learning, International Conference on Learning Representations*, 2021.

[300] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *International Joint Conference on Artificial Intelligence (IJCAI)*, page 4732–4738, 2019.

[301] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv preprint*, 2020.

[302] Thomas Humphries, Simon Oya, Lindsey Tulloch, Matthew Rafuse, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Investigating membership inference attacks under data dependencies. In *IEEE Computer Security Foundations Symposium (CSF)*, pages 473–488, 2023.

[303] Dayong Ye, Sheng Shen, Tianqing Zhu, Bo Liu, and Wanlei Zhou. One Parameter Defense—Defending Against Data Inference Attacks via Differential Privacy. *IEEE Transactions on Information Forensics and Security (TIFS)*, 17:1466–1480, 2022.

[304] Pierre Stock, Igor Shilov, Ilya Mironov, and Alexandre Sablayrolles. Defending against Reconstruction attacks using Rényi Differential Privacy. `https://openreview.net/forum?id=e0GcQ9l4Dh`, 2023.

[305] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers. *Int. J. Secur. Netw.*, 10(3):137–150, September 2015.

[306] Maria S. Esipova, Atiyeh Ashari Ghomi, Yaqiao Luo, and Jesse C Cresswell. Disparate Impact in Differential Privacy from Gradient Misalignment. In *International Conference on Learning Representations (ICLR)*, 2023.

[307] Cuong Tran, My H Dinh, and Ferdinando Fioretto. Differentially Private Empirical Risk Minimization under the Fairness Lens. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[308] Bo Zhang, Ruotong Yu, Haipei Sun, Yanying Li, Jun Xu, and Hui Wang. Privacy for all: Demystify vulnerability disparity of differential privacy against membership inference attack. *arXiv preprint*, 2020.

[309] David Solans, Battista Biggio, and Carlos Castillo. Poisoning Attacks on Algorithmic Fairness. In *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD)*, page 162–177, 2020.

[310] Minh-Hao Van, Wei Du, Xintao Wu, and Aidong Lu. Poisoning attacks on fair machine learning. In *International Conference on Database Systems for Advanced Applications*, pages 370–386, 2022.

[311] Ninareh Mehrabi, Muhammad Naveed, Fred Morstatter, and Aram Galstyan. Exacerbating Algorithmic Bias through Fairness Attacks. *AAAI Conference on Artificial Intelligence*, 35(10):8930–8938, May 2021.

[312] Huan Tian, Guangsheng Zhang, Bo Liu, Tianqing Zhu, Ming Ding, and Wanlei Zhou. FD-MIA: Efficient Attacks on Fairness-enhanced Models. *arXiv preprint*, 2023.

[313] Julien Ferry, Ulrich Aïvodji, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. Exploiting Fairness to Enhance Sensitive Attributes Reconstruction. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 18–41, 2023.

[314] Faisal Kamiran and Toon Calders. Data Pre-Processing Techniques for Classification without Discrimination. *Knowledge and Information Systems*, 33, 2011.

[315] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. A Reductions Approach to Fair Classification. In *International Conference on Machine Learning (ICML)*, pages 60–69, 2018.

[316] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[317] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable Deep Learning under Fire. In *USENIX Security Symposium (USENIX SEC)*, 2020.

[318] Pengrui Quan, Supriyo Chakraborty, Jeya Vikranth Jeyakumar, and Mani Srivastava. On the amplification of security and privacy risks by post-hoc explanations in machine learning models. *arXiv preprint*, 2022.

[319] Hengtong Zhang, Jing Gao, and Lu Su. Data Poisoning Attacks Against Outcome Interpretations of Predictive Models. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, page 2165–2173, 2021.

[320] Hubert Baniecki, Wojciech Kretowicz, and Przemyslaw Biecek. Fooling Partial Dependence via Data Poisoning. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 121–136, 2023.

[321] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[322] Dylan Z Slack, Sophie Hilgard, Himabindu Lakkaraju, and Sameer Singh. Counterfactual Explanations Can Be Manipulated. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[323] Yongjie Wang, Hangwei Qian, and Chunyan Miao. DualCF: Efficient Model Extraction Attack from Counterfactual Explanations. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 1318–1329, 2022.

[324] Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. Model extraction from counterfactual explanations. *arXiv preprint*, 2020.

[325] Martin Pawelczyk, Himabindu Lakkaraju, and Seth Neel. On the privacy risks of algorithmic recourse. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 9680–9696, 2023.

[326] Han Liu, Yuhao Wu, Zhiyuan Yu, and Ning Zhang. Please Tell Me More: Privacy Impact of Explainability through the Lens of Membership Inference Attack. In *IEEE Symposium on Security and Privacy (S&P)*, pages 120–120, 2024.

[327] Xuejun Zhao, Wencan Zhang, Xiaokui Xiao, and Brian Lim. Exploiting explanations for model inversion attacks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 682–692, 2021.

[328] Aparna Balagopalan, Haoran Zhang, Kimia Hamidieh, Thomas Hartvigsen, Frank Rudzicz, and Marzyeh Ghassemi. The Road to Explainability is Paved with Bias: Measuring the Fairness of Explanations. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 1194–1206, 2022.

[329] Jessica Dai, Sohini Upadhyay, Ulrich Aivodji, Stephen H. Bach, and Himabindu Lakkaraju. Fairness via Explanation Quality: Evaluating Disparities in the Quality of Post Hoc Explanations. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 203–214, 2022.

[330] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9078–9086, 2019.

[331] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *USENIX Security Symposium (USENIX SEC)*, pages 321–338, 2019.

[332] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[333] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P. Gummadi. Fairness Constraints: Mechanisms for Fair Classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 962–970, 2017.

[334] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features through Propagating Activation Differences. In *International Conference on Machine Learning (ICML)*, page 3145–3153, 2017.

[335] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[336] Xiangtao Meng, Tianshuo Cong, Li Wang, Wenyu Chen, Zheng Li, Shanqing Guo, and Xiaoyun Wang. From Defender to Devil? Unintended Risk Interactions Induced by LLM Defenses. *arXiv preprint*, 2025.

[337] Mohammad Yaghini, Patty Liu, Franziska Boenisch, and Nicolas Papernot. Learning with Impartiality to Walk on the Pareto Frontier of Fairness, Privacy, and Utility. *NeurIPS Workshop on Regulatable ML: Towards Bridging the Gaps between Machine Learning Research and Regulations*, 2023.

[338] Shiji Xin, Yifei Wang, Jingtong Su, and Yisen Wang. On the connection between invariant learning and adversarial training for out-of-distribution generalization. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 10519–10527, 2023.

[339] Shu Hu, Zhenhuan Yang, Xin Wang, Yiming Ying, and Siwei Lyu. Outlier Robust Adversarial Training. *arXiv preprint*, 2023.

[340] Zhiqi Bu, Ping Li, and Weijie Zhao. Practical Adversarial Training with Differential Privacy for Deep Learning. `https://openreview.net/forum?id=1hw-h1C8bch`, 2022.

[341] Jiapeng Wu, Atiyeh Ashari Ghomi, David Glukhov, Jesse C. Cresswell, Franziska Boenisch, and Nicolas Papernot. Augment then Smooth: Reconciling Differential Privacy with Certified Robustness. *Transactions on Machine Learning Research (TMLR)*, 2024.

[342] Zhiqi Bu and Yuan Zhang. Differentially Private Optimizers Can Learn Adversarially Robust Models. *Transactions on Machine Learning Research (TMLR)*, 2023.

[343] Fengxiang He, Shaopeng Fu, Bohan Wang, and Dacheng Tao. Robustness, privacy, and generalization of adversarial training. *arXiv preprint*, 2020.

[344] Seyed Ali Mousavi, Hamid Mousavi, and Masoud Daneshtalab. FARMUR: Fair Adversarial Retraining to Mitigate Unfairness in Robustness. In *Advances in Databases and Information Systems (ADBIS)*, pages 133–145, 2023.

[345] Xinsong Ma, Zekai Wang, and Weiwei Liu. On the Tradeoff Between Robustness and Fairness. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 26230–26241, 2022.

[346] Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P Dickerson. Fairness through robustness: Investigating robustness disparity in deep learning. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 466–477, 2021.

[347] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. To be Robust or to be Fair: Towards Fairness in Adversarial Training. In *International Conference on Machine Learning (ICML)*, pages 11492–11501, 2021.

[348] Haipei Sun, Kun Wu, Ting Wang, and Wendy Hui Wang. Towards Fair and Robust Classification. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 356–376, 2022.

[349] Boqi Li and Weiwei Liu. WAT: improve the worst-class robustness in adversarial training. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 14982–14990, 2023.

[350] Hyungyu Lee, Saehyung Lee, Hyemi Jang, Junsung Park, Ho Bae, and Sungroh Yoon. DAFA: Distance-Aware Fair Adversarial Training. In *International Conference on Learning Representations (ICLR)*, 2024.

[351] Zeming Wei, Yifei Wang, Yiwen Guo, and Yisen Wang. Cfa: Class-wise calibrated fair adversarial training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8193–8201, 2023.

[352] Deepak P and Savitha Abraham. FairLOF: Fairness in Outlier Detection. *Data Science and Engineering*, 6, 12 2021.

[353] Haoyu Liu, Fenglong Ma, Shibo He, Jiming Chen, and Jing Gao. Fairness-aware outlier ensemble. *arXiv preprint*, 2021.

[354] Hongjing Zhang and Ian Davidson. Towards fair deep anomaly detection. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 138–148, 2021.

[355] Cuong Tran, Ferdinando Fioretto, and Pascal Van Hentenryck. Differentially private and fair deep learning: A lagrangian dual approach. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 9932–9939, 2021.

[356] Wenyan Liu, Xiangfeng Wang, Haikun Zheng, Bo Jin, Xiaoling Wang, and Hongyuan Zha. Mitigating disparate impact on model accuracy in differentially private learning. *Information Sciences*, 616:108–126, 2022.

[357] Andrew Lowy, Devansh Gupta, and Meisam Razaviyayn. Stochastic Differentially Private and Fair Learning. In *International Conference on Learning Representations (ICLR)*, pages 86–119, 2023.

[358] Matthew Jagielski, Michael Kearns, Jieming Mao, Alina Oprea, Aaron Roth, Saeed Sharifi Malvajerdi, and Jonathan Ullman. Differentially Private Fair Learning. In *International Conference on Machine Learning (ICML)*, pages 3000–3008, 2019.

[359] Mohammad Yaghini, Patty Liu, Franziska Boenisch, and Nicolas Papernot. Regulation Games for Trustworthy Machine Learning. *NeurIPS Workshop on Regulatable ML: Towards Bridging the Gaps between Machine Learning Research and Regulations*, 2023.

[360] Jiahao Ding, Xinyue Zhang, Xiaohuan Li, Junyi Wang, Rong Yu, and Miao Pan. Differentially private and fair classification via calibrated functional mechanism. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 622–629, 2020.

[361] Depeng Xu, Shuhan Yuan, and Xintao Wu. Achieving Differential Privacy and Fairness in Logistic Regression. In *World Wide Web Conference (WWW)*, page 594–599, 2019.

[362] Tao Zhang, Tianqing Zhu, Kun Gao, Wanlei Zhou, and S Yu Philip. Balancing learning model privacy, fairness, and accuracy with early stopping criteria. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2021.

[363] Depeng Xu, Wei Du, and Xintao Wu. Removing Disparate Impact on Model Accuracy in Differentially Private Stochastic Gradient Descent. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, page 1924–1932, 2021.

[364] Khang Tran, Ferdinando Fioretto, Issa Khalil, My T. Thai, Linh Thi Xuan Phan, and NhatHai Phan. FairDP: Achieving Fairness Certification with Differential Privacy . In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 956–976, 2025.

[365] Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. Robust and stable black box explanations. In *International Conference on Machine Learning (ICML)*, pages 5628–5638, 2020.

[366] Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. Robust Attribution Regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[367] Tang Li, Fengchun Qiao, Mengmeng Ma, and Xi Peng. Are Data-Driven Explanations Robust Against Out-of-Distribution Data? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3821–3831, 2023.

[368] Fan Yang, Qizhang Feng, Kaixiong Zhou, Jiahao Chen, and Xia Hu. Differentially private counterfactuals via functional mechanism. *arXiv preprint*, 2022.

[369] NhatHai Phan, Minh Vu, Yang Liu, Ruoming Jin, Dejing Dou, Xintao Wu, and My T Thai. Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness. *arXiv preprint*, 2019.

[370] Hai Phan, My T Thai, Han Hu, Ruoming Jin, Tong Sun, and Dejing Dou. Scalable differential privacy with certified robustness in adversarial learning. In *International Conference on Machine Learning (ICML)*, pages 7683–7694, 2020.

[371] Neel Patel, Reza Shokri, and Yair Zick. Model Explanations with Differential Privacy. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 1895–1904, 2022.

[372] Julien Ferry, Ulrich Aïvodji, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. SoK: Taming the Triangle - On the Interplays between Fairness, Interpretability and Privacy in Machine Learning. In *https://hal.science/hal-04359832*, 2023.

[373] Janvi Thakkar, Giulio Zizzo, and Sergio Maffeis. Elevating Defenses: Bridging Adversarial Training and Watermarking for Model Resilience. *Deployable AI@ AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[374] Chang Xu, Jun Wang, Francisco Guzmán, Benjamin Rubinstein, and Trevor Cohn. Mitigating Data Poisoning in Text Classification with Differential Privacy. In *Findings of the Empirical Methods in Natural Language Processing (EMNLP Findings)*, pages 4348–4356, 2021.

[375] Daniël Vos, Jelle Vos, Tianyu Li, Zekeriya Erkin, and Sicco Verwer. Differentially-Private Decision Trees with Probabilistic Robustness to Data Poisoning. *arXiv preprint*, 2023.

[376] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2019.

[377] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Robustness may be at odds with fairness: An empirical study on class-wise accuracy. In *NeurIPS 2020 Workshop on pre-registration in machine learning*, pages 325–342, 2021.

[378] Jamie Hayes, Borja Balle, and M Pawan Kumar. Learning to be adversarially robust and differentially private. *arXiv preprint*, 2022.

[379] Prasad Chalasani, Jiefeng Chen, Amrita Roy Chowdhury, Xi Wu, and Somesh Jha. Concise Explanations of Neural Networks using Adversarial Training. In *International Conference on Machine Learning (ICML)*, pages 1383–1391, 2020.

[380] Rachel Cummings, Varun Gupta, Dhamma Kimpara, and Jamie Morgenstern. On the Compatibility of Privacy and Fairness. In *Adjunct Publication of the Conference on User Modeling, Adaptation and Personalization (UMAP)*, page 309–315, 2019.

[381] Sushant Agarwal. Trade-Offs between Fairness and Privacy in Machine Learning. In *IJCAI Workshop on AI for Social Good*, 2021.

[382] Hussein Mozannar, Mesrob Ohannessian, and Nathan Srebro. Fair learning with private demographic data. In *International Conference on Machine Learning (ICML)*, pages 7066–7075, 2020.

[383] David Banisar. The Right to Information and Privacy: Balancing Rights and Managing Conflicts. *SSRN Electronic Journal*, 03 2011.

[384] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[385] Mintong Kang, Dawn Song, and Bo Li. Diffattack: Evasion attacks against diffusion-based adversarial purification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[386] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, pages 1–47, 2022.

[387] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, pages 2206–2216, 2020.

[388] Maura Pintor, Luca Demetrio, Angelo Sotgiu, Marco Melis, Ambra Demontis, and Battista Biggio. Secml: Secure and explainable machine learning in Python. *SoftwareX*, 18:101095, 2022.

[389] Maura Pintor, Luca Demetrio, Angelo Sotgiu, Ambra Demontis, Nicholas Carlini, Battista Biggio, and Fabio Roli. Indicators of Attack Failure: Debugging and Improving Optimization of Adversarial Examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 23063–23076, 2022.

[390] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *arXiv preprint*, 2019.

[391] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM workshop on artificial intelligence and security (AISec)*, pages 3–14, 2017.

[392] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7:47230–47244, 2019.

[393] Zeyang Sha, Xinlei He, Pascal Berrang, Mathias Humbert, and Yang Zhang. Fine-tuning is all you need to mitigate backdoor attacks. *arXiv preprint*, 2022.

[394] Yiming Li, Ziqi Zhang, Jiawang Bai, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Open-sourced dataset protection via backdoor watermarking. *NeurIPS Workshop on Dataset Curation and Security*, 2020.

[395] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint*, 2021.

[396] Xiaotian Han, Jianfeng Chi, Yu Chen, Qifan Wang, Han Zhao, Na Zou, and Xia Hu. FFB: A Fair Fairness Benchmark for In-Processing Group Fairness Methods. In *International Conference on Learning Representations (ICLR)*, 2024.

[397] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint*, 2020.

[398] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A Closer Look at Accuracy vs. Robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8588–8601, 2020.

[399] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (proper) definition. In *International Conference on Machine Learning (ICML)*, pages 17258–17277, 2022.

[400] Jiayuan Ye, Zhenyu Zhu, Fanghui Liu, Reza Shokri, and Volkan Cevher. Initialization Matters: Privacy-Utility Analysis of Overparameterized Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5419–5446, 2023.

[401] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 9312–9321, 2021.

[402] Florian Tramèr and Dan Boneh. Differentially Private Learning Needs Better Features (or Much More Data). In *International Conference on Learning Representations (ICLR)*, 2020.

[403] Honglin Fu et al. Perseus: Tracing the Masterminds Behind Cryptocurrency Pump-and-Dump Schemes. In *arXiv preprint arXiv:2503.01686*, 2025.

[404] Halcyon. What is an Initial Access Broker (IAB)? Accessed 2026-05-18.

[405] Yugeng Liu, Zheng Li, Hai Huang, Michael Backes, and Yang Zhang. Amplifying Machine Learning Attacks Through Strategic Compositions. *arXiv preprint*, 2025.

[406] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision (ECCV)*, pages 484–501, 2020.

[407] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*, 2019.

[408] Peng Yifeng, Zhizheng Wu, and Chen Chen. Swallowing the Poison Pills: Insights from Vulnerability Disparity Among LLMs. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 38–55, 2025.

[409] Xinli Yue, Ningping Mou, Qian Wang, and Lingchen Zhao. Revisiting adversarial training under long-tailed distributions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24492–24501, 2024.

[410] Seungju Cho, Hongsin Lee, and Changick Kim. Long-tailed Adversarial Training with Self-Distillation. In *International Conference on Learning Representations (ICLR)*, 2025.

[411] Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-Only Membership Inference Attacks. In *International Conference on Machine Learning (ICML)*, pages 1964–1974, 2021.

[412] Hao Li, Zheng Li, Siyuan Wu, Yutong Ye, Min Zhang, Dengguo Feng, and Yang Zhang. Enhanced label-only membership inference attacks with fewer queries. In *USENIX Security Symposium (USENIX SEC)*, 2025.

[413] Ziang Li, Hongguang Zhang, Juan Wang, Meihui Chen, Hongxin Hu, Wenzhe Yi, Xiaoyang Xu, Mengda Yang, and Chenjun Ma. From head to tail: efficient black-box model inversion attack via long-tailed learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29288–29298, 2025.

[414] Jiayuan Ye, Anastasia Borovykh, Soufiane Hayou, and Reza Shokri. Leave-one-out Distinguishability in Machine Learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[415] Yi Xie, Jie Zhang, Shiqian Zhao, Tianwei Zhang, and Xiaofeng Chen. Same: Sample reconstruction against model extraction attacks. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 19974–19982, 2024.

[416] Ambra Demontis et al. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *USENIX Security Symposium (USENIX SEC)*, pages 321–338, 2019.

[417] Lei Wu and Zhanxing Zhu. Towards Understanding and Improving the Transferability of Adversarial Examples in Deep Neural Networks. In *Asian Conference on Machine Learning (ACML)*, pages 837–850, 2020.

[418] Jindong Gu, Xiaojun Jia, Pau de Jorge, Wenqian Yu, Xinwei Liu, Avery Ma, Yuan Xun, Anjun Hu, Ashkan Khakzar, Zhijiang Li, Xiaochun Cao, and Philip Torr. A Survey on Transferability of Adversarial Examples Across Deep Neural Networks. *Transactions on Machine Learning Research (TMLR)*, 2024.

[419] Yuxin Wen, Arpit Bansal, Hamid Kazemi, Eitan Borgnia, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Canary in a Coalmine: Better Membership Inference with Ensembled Adversarial Queries. In *International Conference on Learning Representations (ICLR)*, 2023.

[420] Shagufta Mehnaz, Sayanton V. Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. Are Your Sensitive Attributes Private? Novel Model Inversion Attribute Inference Attacks on Classification Models. In *USENIX Security Symposium (USENIX SEC)*, pages 4579–4596, 2022.

[421] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 36–52, 2018.

[422] Seong Joon Oh, Max Augustin, Mario Fritz, and Bernt Schiele. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[423] Wenbo Jiang, Hongwei Li, Yuxin Lu, Wenshu Fan, and Rui Zhang. Adversarial Robustness Poisoning: Increasing Adversarial Vulnerability of the Model via Data Poisoning. In *IEEE Global Communications Conference (GLOBECOM)*, pages 4286–4291, 2024.

[424] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 85–99, 2020.

[425] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:30339–30351, 2021.

[426] Zitao Chen and Karthik Pattabiraman. A method to facilitate membership inference attacks in deep learning models. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

[427] Yuxin Wen, Leo Marchyok, Sanghyun Hong, Jonas Geiping, Tom Goldstein, and Nicholas Carlini. Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 83374–83396, 2024.

[428] Yanjun Zhang, Guangdong Bai, Mahawaga Arachchige Pathum Chamikara, Mengyao Ma, Liyue Shen, Jingwei Wang, Surya Nepal, Minhui Xue, Long Wang, and Joseph Liu. AgrEvader: Poisoning Membership Inference against Byzantine-robust Federated Learning. In *ACM Web Conference (WWW)*, page 2371–2382, 2023.

[429] Mohammad Malekzadeh, Anastasia Borovykh, and Deniz Gündüz. Honest-but-Curious Nets: Sensitive Attributes of Private Inputs Can Be Secretly Coded into the Classifiers' Outputs. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 825–844, 2021.

[430] Ruyi Ding, Shijin Duan, Xiaolin Xu, and Yunsi Fei. VertexSerum: Poisoning Graph Neural Networks for Link Inference. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4509–4518, 2023.

[431] Zhibo Wang, Yuting Huang, Mengkai Song, Libing Wu, Feng Xue, and Kui Ren. Poisoning-Assisted Property Inference Attack Against Federated Learning. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 20(4):3328–3340, 2023.

[432] Yulong Tian, Fnu Suya, Anshuman Suri, Fengyuan Xu, and David Evans. Manipulating transfer learning for property inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15975–15984, 2023.

[433] Xinjian Luo, Yangfan Jiang, Fei Wei, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Exploring privacy and fairness risks in sharing diffusion models: An adversarial perspective. *IEEE Transactions on Information Forensics and Security (TIFS)*, 19:8109–8124, 2024.

[434] Ashwinee Panda, Christopher A. Choquette-Choo, Zhengming Zhang, Yaoqing Yang, and Prateek Mittal. Teach LLMs to Phish: Stealing Private Information from Language Models. In *International Conference on Learning Representations (ICLR)*, 2024.

[435] Shanglun Feng and Florian Tramèr. Privacy Backdoors: Stealing Data with Corrupted Pretrained Models. In *International Conference on Machine Learning (ICML)*, 2024.

[436] Yuefeng Peng, Junda Wang, Hong Yu, and Amir Houmansadr. Data extraction attacks in retrieval-augmented generation via backdoors. *arXiv preprint*, 2024.

[437] Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation . In *IEEE Symposium on Security and Privacy (S&P)*, pages 1287–1305, 2024.

[438] Nicolas Küchler, Ivan Petrov, Conrad Grobler, and Ilia Shumailov. Architectural backdoors for within-batch data stealing and model inference manipulation. *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026.

[439] Ali Naseh, Jaechul Roh, Eugene Bagdasarian, and Amir Houmansadr. Backdooring bias (B2) into stable diffusion models. In *USENIX Security Symposium (USENIX SEC)*, 2025.

[440] Syed Irfan Ali Meerza and Jian Liu. EAB-FL: exacerbating algorithmic bias through model poisoning attacks in federated learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[441] Nicholas Furth, Abdallah Khreishah, Guanxiong Liu, NhatHai Phan, and Yaser Jararweh. Un-Fair Trojan: Targeted Backdoor Attacks Against Model Fairness. In *International Conference on Software Defined Systems (SDS)*, pages 1–9, 2022.

[442] Jiaqi Xue, Mengxin Zheng, Yi Sheng, Lei Yang, Qian Lou, and Lei Jiang. TrojFair: Trojan Fairness Attacks. In *ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis (LAMPS)*, page 47–56, 2024.

[443] Linlin Wang, Tianqing Zhu, Laiqiao Qin, Longxiang Gao, and Wanlei Zhou. Bias amplification in rag: Poisoning knowledge retrieval to steer llms. *arXiv preprint*, 2025.

[444] Benjamin Zi Hao Zhao, Aviral Agrawal, Catisha Coburn, Hassan Jameel Asghar, Raghav Bhaskar, Mohamed Ali Kaafar, Darren Webb, and Peter Dickinson. On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models . In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 232–251, 2021.

[445] Yaxin Xiao, Qingqing Ye, Haibo Hu, Huadi Zheng, Chengfang Fang, and Jie Shi. MExMI: Pool-based Active Model Extraction Crossover Membership Inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10203–10216, 2022.

[446] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Guanghao Mei, and Qian Wang. InverseNet: Augmenting Model Extraction Attacks with Training Data Inversion. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2439–2447, 2021. Main Track.

[447] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, page 506–519, 2017.

[448] Chen Chen, Xuanli He, Lingjuan Lyu, and Fangzhao Wu. Killing one bird with two stones: Model extraction and attribute inference attacks against bert-based apis. *arXiv preprint*, 2021.

[449] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of Dataset Properties in Multi-Party Machine Learning. In *USENIX Security Symposium (USENIX SEC)*, pages 2687–2704, 2021.

[450] Hao Fang, Bin Chen, Xuan Wang, Zhi Wang, and Shu-Tao Xia. Gifd: A generative gradient inversion method with feature domain optimization. In *International Conference on Computer Vision (ICCV)*, pages 4967–4976, 2023.

[451] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *USENIX Security Symposium (USENIX SEC)*, pages 515–532, 2019.

[452] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In *USENIX Security Symposium (USENIX SEC)*, pages 2003–2020, 2020.

[453] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *Annual Design Automation Conference (DAC)*, 2018.

[454] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips. In *USENIX Security Symposium (USENIX SEC)*, pages 1463–1480, 2020.

[455] Adnan Siraj Rakin, Yukui Luo, Xiaolin Xu, and Deliang Fan. Deep-Dup: An Adversarial Weight Duplication Attack Framework to Crush Deep Neural Network in Multi-Tenant FPGA. In *USENIX Security Symposium (USENIX SEC)*, pages 1919–1936, 2021.

[456] Yukui Luo, Cheng Gongye, Yunsi Fei, and Xiaolin Xu. DeepStrike: Remotely-Guided Fault Injection Attacks on DNN Accelerator in Cloud-FPGA. In *Annual ACM/IEEE Design Automation Conference (DAC)*, page 295–300, 2022.

[457] Kathrin Grosse, Thomas A. Trost, Marius Mosbach, and Michael Backes. Adversarial Initialization - when your network performs the way I want -. *arXiv preprint*, February 2019.

[458] Kathrin Grosse, Thomas A. Trost, Marius Mosbach, Michael Backes, and Dietrich Klakow. On the Security Relevance of Initial Weights in Deep Neural Networks. In *International Conference on Artificial Neural Networks*, page 3–14, 2020.

[459] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross J Anderson. Manipulating sgd with data ordering attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:18021–18032, 2021.

[460] Qixuan Jin, Walter Gerych, Abinitha Gourabathina, Vinith Menon Suriyakumar, and Marzyeh Ghassemi. TOGA: Trigger Optimization for Clean Data Ordering Backdoor Attack. https://openreview.net/forum?id=NbyhwSvOgE, 2026.

[461] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge Examples: Energy-Latency Attacks on Neural Networks. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 212–231, 2021.

[462] Antonio Emanuele Cinà, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. Energy-latency attacks via sponge poisoning. *Information Sciences*, 702:121905, 2025.

[463] Zijian Wang, Shuo Huang, Yujin Huang, and Helei Cui. Energy-Latency Attacks to On-Device Neural Networks via Sponge Poisoning. In *Secure and Trustworthy Deep Learning Systems Workshop (SecTL)*, 2023.

[464] Ezekiel Soremekun, Sakshi Udeshi, and Sudipta Chattopadhyay. Towards backdoor attacks and defense in robust machine learning models. *Computers & Security*, 127:103101, 2023.

[465] Qi Zhao and Christian Wressnegger. Adversarially Robust Anti-Backdoor Learning. In *Workshop on Artificial Intelligence and Security*, page 77–88, 2024.

[466] Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. Data Cards: Purposeful and Transparent Dataset Documentation for Responsible AI. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 1776–1826, 2022.

[467] HuggingFace. Dataset Cards — huggingface.co. *https://huggingface.co/docs/hub/en/datasets-cards*, 2024. [Accessed 11-05-2024].

[468] Google. Google Cloud Model Cards — modelcards.withgoogle.com. *https://modelcards.withgoogle.com/*, 2024. [Accessed 11-04-2024].

[469] HuggingFace. Model Cards — huggingface.co. *https://huggingface.co/docs/hub/en/model-cards*, 2024. [Accessed 11-04-2024].

[470] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 2968–2985, 2021.

[471] Chenyu Huang, Jianzong Wang, Huangxun Chen, Shijing Si, Zhangcheng Huang, and Jing Xiao. zkMLaaS: a Verifiable Scheme for Machine Learning as a Service. In *IEEE Global Communications Conference (GLOBECOM)*, pages 5475–5480, 2022.

[472] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. ZEN: An Optimizing Compiler for Verifiable, Zero-Knowledge Neural Network Inferences. Cryptology ePrint Archive, Paper 2021/087, 2021. https://eprint.iacr.org/2021/087.

[473] Haochen Sun, Jason Li, and Hongyang Zhang. zkLLM: Zero Knowledge Proofs for Large Language Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 4405–4419, 2024.

[474] Yehuda Lindell. Secure multiparty computation. *Commun. ACM*, 64(1):86–96, December 2020.

[475] Shahla Atapoor, Nigel P. Smart, and Younes Talibi Alaoui. Private Liquidity Matching Using MPC. In *Topics in Cryptology – CT-RSA 2022*, pages 96–119, 2022.

[476] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. CRYPTEN: secure multi-party computation meets machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[477] MPC-Alliance. MPC Alliance. `https://www.mpcalliance.org`, 2023.

[478] Darya Kaviani and Raluca Ada Popa. MPC Deployments. `https://mpc.cs.berkeley.edu`, 2023.

[479] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. Cerebro: A Platform for Multi-Party Cryptographic Collaborative Learning. In *USENIX Security Symposium (USENIX SEC)*, pages 2723–2740, 2021.

[480] Lachlan J Gunn, N Asokan, Jan-Erik Ekberg, Hans Liljestrand, and Vijayanand Prasad. *Hardware Platform Security for Mobile Devices*. Springer, 2022. `https://ssg.aalto.fi/publications/hardware-platform-security-for-mobile-devices/`.

[481] Ahmad-Reza Sadeghi and Christian Stüble. Property-Based Attestation for Computing Platforms: Caring about Properties, Not Mechanisms. In *Workshop on New Security Paradigms*, 2004.

[482] Kari Kostiainen, N. Asokan, and Jan-Erik Ekberg. Practical Property-Based Attestation on Mobile Devices. In *Trust and Trustworthy Computing*, pages 78–92, 2011.

[483] Intel. Intel Software Guard Extensions Programming Reference. Documentation, Intel, 2014.

[484] High-Level Expert Group on AI. Ethics guidelines for trustworthy AI. Report, European Commission, Brussels, April 2019.

[485] Elham Tabassi, Kevin J. Burns, M. Hadjimichael, Andres Molina-Markham, and Julian Sexton. A Taxonomy and Terminology of Adversarial Machine Learning. NIST Interagency/Internal Report, NIST, 2019.

[486] Iñigo de Miguel Beriain, Pilar Nicolás Jiménez, María José Rementería, Davide Cirillo, Atia Cortés, Diego Saby, and Guillermo Lazcoz Moratinos. Auditing the quality of datasets used in algorithmic decision-making systems. Technical report, European Parliamentary Research Service, 2022. PE 729.541.

[487] U.S. Congress. H.R. 6580 – Algorithmic Accountability Act. *https://www.congress.gov/bill/117th-congress/house-bill/6580/text*, 2022.

[488] European Commission. Regulation (EU) 2024/1689 (Artificial Intelligence Act). *https://artificialintelligenceact.eu/*, 2024.

[489] European Union Law. Art. 35 GDPR Data protection impact assessment. *General Data Protection Regulation (GDPR)*, 2018.

[490] Emre Kazim, Danielle Mendes Thame Denny, and Adriano Koshiyama. AI auditing and impact assessment: according to the UK information commissioner's office. *AI and Ethics*, Feb 2021.

[491] White House. Guidance for Regulation of Artificial Intelligence Applications. *Memorandum For The Heads Of Executive Departments And Agencies*, 2020.

[492] Amanda Coston, Anna Kawakami, Haiyi Zhu, Ken Holstein, and Hoda Heidari. A Validity Perspective on Evaluating the Justified Use of Data-driven Decision-making Algorithms. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 690–704, 2023.

[493] Yuantian Miao, Minhui Xue, Chao Chen, Lei Pan, Jun Zhang, Benjamin Zi Hao Zhao, Dali Kaafar, and Yang Xiang. The Audio Auditor: User-Level Membership Inference in Internet of Things Voice Services. In *Privacy Enhancing Technology Symposium (PETS)*, 2021.

[494] Congzheng Song and Vitaly Shmatikov. Auditing Data Provenance in Text-Generation Models. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, page 196–206, 2019.

[495] Xiao Liu and Sotirios A Tsaftaris. Have you forgotten? A method to assess if machine learning models have forgotten data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 95–105, 2020.

[496] Yangsibo Huang, Xiaoxiao Li, and Kai Li. EMA: Auditing Data Removal from Trained Models. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2021.

[497] Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1039–1056, 2021.

[498] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning. In *USENIX Security Symposium (USENIX SEC)*, pages 501–518, 2021.

[499] Xiaokuan Zhang, Yang Zhang, and Yinqian Zhang. VeriTrain: Validating MLaaS Training Efforts via Anomaly Detection. *IEEE Transactions on Dependable and Secure Computing*, pages 1–17, 2023.

[500] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. SafetyNets: Verifiable Execution of Deep Neural Networks on an Untrusted Cloud. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 4675–4684, 2017.

[501] Haochen Sun and Hongyang Zhang. zkDL: Efficient Zero-Knowledge Proofs of Deep Learning Training. Cryptology ePrint Archive, Paper 2023/1174, 2023. https://eprint.iacr.org/2023/1174.

[502] Ali Shahin Shamsabadi, Sierra Calanda Wyllie, Nicholas Franzese, Natalie Dullerud, Sébastien Gambs, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-PROFITT: Confidential PROof of FaIr Training of Trees. In *International Conference on Learning Representations (ICLR)*, 2023.

[503] Sanjam Garg, Aarushi Goel, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Guru-Vamsi Policharla, and Mingyuan Wang. Experimenting with Zero-Knowledge Proofs of Training. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 1880–1894, 2023.

[504] Kasra Abbaszadeh, Christodoulos Pappas, Dimitrios Papadopoulos, and Jonathan Katz. Zero-Knowledge Proofs of Training for Deep Neural Networks. Cryptology ePrint Archive, Paper 2024/162, 2024. https://eprint.iacr.org/2024/162.

[505] Thorsten Eisenhofer, Doreen Riepel, Varun Chandrasekaran, Esha Ghosh, Olga Ohrimenko, and Nicolas Papernot. Verifiable and Provably Secure Machine Unlearning . In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 479–496, 2025.

[506] Saerom Park, Seongmin Kim, and Yeon-sup Lim. Fairness Audit of Machine Learning Models with Confidential Computing. In *ACM Web Conference (WWW)*, page 3488–3499, 2022.

[507] Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. Fairness in the Eyes of the Data: Certifying Machine-Learning Models. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 926–935, 2021.

[508] Chhavi Yadav, Amrita Roy Chowdhury, Dan Boneh, and Kamalika Chaudhuri. FairProof : Confidential and Certifiable Fairness for Neural Networks. In *International Conference on Machine Learning (ICML)*, pages 55682–55705, 2024.

[509] Ali Shahin Shamsabadi, Gefei Tan, Tudor Ioan Cebere, Aurélien Bellet, Hamed Haddadi, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-DPproof: Confidential Proof of Differentially Private Training. In *International Conference on Learning Representations (ICLR)*, 2024.

[510] Payman Mohassel and Yupeng Zhang. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 19–38, 2017.

[511] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation. In *USENIX Security Symposium (USENIX SEC)*, pages 2165–2182, 2021.

[512] Marcel Keller and Ke Sun. Secure quantized training for deep learning. In *International Conference on Machine Learning (ICML)*, pages 10912–10938, 2022.

[513] Payman Mohassel and Peter Rindal. ABY3: A Mixed Protocol Framework for Machine Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 35–52, 2018.

[514] Lucien K. L. Ng and Sherman S. M. Chow. SoK: Cryptographic Neural-Network Computation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 497–514, 2023.

[515] Hidde Lycklama, Alexander Viand, Nicolas Küchler, Christian Knabenhans, and Anwar Hithnawi. Holding Secrets Accountable: Auditing Privacy-Preserving Machine Learning. In *USENIX Security Symposium (USENIX SEC)*, page 991–1008, 2024.

[516] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A Hybrid Secure Computation

Framework for Machine Learning Applications. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, page 707–721, 2018.

[517] Steven Englehardt. Next steps in privacy-preserving telemetry with Prio, 2019.

[518] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[519] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[520] Shahbaz Rezaei and Xin Liu. On the Difficulty of Membership Inference Attacks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7892–7900, 2021.

[521] Zhifeng Kong, Amrita Roy Chowdhury, and Kamalika Chaudhuri. Forgeability and Membership Inference Attacks. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, page 25–31, 2022.

[522] Muhammad Usama Sardar and Christof Fetzer. Confidential computing and related technologies: a critical review. *Cybersecur.*, 6(1):10, 2023.

[523] Intel. Accelerate AI Workloads with Intel Advanced Matrix Extensions (Intel AMX). Solution brief, Intel, 2024.

[524] Nvidia. NVIDIA H100 Tensor Core GPU. Datasheet, Nvidia, 2024.

[525] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. PPFL: privacy-preserving federated learning with trusted execution environments. In *Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, page 94–108, 2021.

[526] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. DarkneTZ: towards model privacy at the edge using trusted execution environments. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, page 161–174, 2020.

[527] Fan Mo, Zahra Tarkhani, and Hamed Haddadi. Machine Learning with Confidential Computing: A Systematization of Knowledge. *ACM Computing Surveys*, 56(11):281:1–281:40, 2024.

[528] Dmitrii Kuvaiskii, Gaurav Kumar, and Mona Vij. Computation offloading to hardware accelerators in Intel SGX and Gramine library OS. *arXiv preprint*, 2022.

[529] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[530] Ron Kohavi. Census Income. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5GP7S.

[531] Zhifei Zhang, Yang Song, and Hairong Qi. Age Progression/Regression by Conditional Adversarial Autoencoder. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[532] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In *Association for Computational Linguistics (ACL)*, pages 142–150, 2011.

[533] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, 2014.

[534] Stephan Van Schaik, Alex Seto, Thomas Yurek, Adam Batori, Bader AlBassam, Daniel Genkin, Andrew Miller, Eyal Ronen, Yuval Yarom, and Christina Garman. SoK: SGX.Fail: How Stuff Gets eXposed . In *IEEE Symposium on Security and Privacy (S&P)*, pages 4143–4162, 2024.

[535] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *USENIX Security Symposium (USENIX SEC)*, page 991–1008, 2018.

[536] Pietro Borrello, Daniele Cono D'Elia, Leonardo Querzoni, Cristiano Giuffrida, Jong Kim, Giovanni Vigna, and Elaine Shi. Constantine: Automatic Side-Channel Resistance Using Efficient Control and Data Flow Linearization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 715–733, 2021.

[537] Victor Costan, Ilia A. Lebedev, Srinivas Devadas, and Stefan Savage. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *USENIX Security Symposium (USENIX SEC)*, pages 857–874, 2016.

[538] Jiyong Yu, Mengjia Yan, Artem Khyzha, Adam Morrison, Josep Torrellas, and Christopher W. Fletcher. Speculative Taint Tracking (STT): A Comprehensive Protection for Speculatively Accessed Data. *IEEE Micro*, 40(3):81–90, 2020.

[539] Linyi Li, Tao Xie, and Bo Li. SoK: Certified Robustness for Deep Neural Networks . In *IEEE Symposium on Security and Privacy (S&P)*, pages 1289–1310, 2023.

[540] Jiayin Jin, Zeru Zhang, Yang Zhou, and Lingfei Wu. Input-agnostic certified group fairness via gaussian parameter smoothing. In *International Conference on Machine Learning (ICML)*, pages 10340–10361, 2022.

[541] Haitham Khedr and Yasser Shoukry. CertiFair: a framework for certified global fairness of neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[542] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Perfectly Parallel Fairness Certification of Neural Networks. *Proceedings of the ACM on Programming Languages (OOPSLA)*, 4, Nov 2020.

[543] Florian Tramer and Dan Boneh. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. In *International Conference on Learning Representations (ICLR)*, 2019.

[544] Z. Zhang, C. Gong, Y. Cai, Y. Yuan, B. Liu, D. Li, Y. Guo, and X. Chen. No Privacy Left Outside: On the (In-)Security of TEE-Shielded DNN Partition for On-Device ML. In *IEEE Symposium on Security and Privacy (S&P)*, pages 55–55, 2024.

[545] Jianwei Zhu, Hang Yin, Peng Deng, Aline Almeida, and Shunfan Zhou. Confidential Computing on NVIDIA Hopper GPUs: A Performance Benchmark Study. *arXiv preprint*, 2024.

[546] Prach Chantasantitam, Adam Ilyas Caulfield, **Vasisht Duddu**, Lachlan J Gunn, and N Asokan. PAL* M: Property Attestation for Large Generative Models. *arXiv preprint*, 2026.

[547] Milad Nasr, Chawin Sitawarin, Sander V Schulhoff, Michael Ilie, Jamie Hayes, Juliette Pluto, Abhradeep Guha Thakurta, Shuang Song, Andreas Terzis, Ilia Shumailov, Harsh Chaudhari, Kai Yuanqing Xiao, Florian Tramèr, and Nicholas Carlini. the attacker moves second: stronger adaptive attacks bypass defenses against llm jailbreaks and prompt injections. https://openreview.net/forum?id=7B9mTg7z25, 2026.

[548] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, page 79–90, 2023.

[549] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *USENIX Security Symposium (USENIX SEC)*, pages 1831–1847, 2024.

[550] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 82895–82920, 2024.

[551] Yan Shvartzshnaider and Vasisht Duddu. Privacy Bias in Language Models: A Contextual Integrity-based Auditing Metric. *arXiv preprint arXiv:2409.03735*, 2024.

[552] Yan Shvartzshnaider and Vasisht Duddu. Position: Contextual Integrity is Inadequately Applied to Language Models. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025.

[553] László Szekeres, Mathias Payer, Tao Wei, and Dawn Song. SoK: Eternal War in Memory. In *IEEE Symposium on Security and Privacy (S&P)*, pages 48–62, 2013.

[554] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To! In *International Conference on Learning Representations (ICLR)*, 2024.

[555] Teodora Baluta, Shiqi Shen, S. Hitarth, Shruti Tople, and Prateek Saxena. Membership Inference Attacks and Generalization: A Causal Perspective. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 249–262, 2022.

[556] Tasfia Mashiat, Xavier Gitiaux, Huzefa Rangwala, Patrick Fowler, and Sanmay Das. Trade-offs between Group Fairness Metrics in Societal Resource Allocation. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, page 1095–1105, 2022.

[557] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[558] Rishi Bommasani, Dilara Soylu, Thomas I. Liao, Kathleen A. Creel, and Percy Liang. Ecosystem Graphs: Documenting the Foundation Model Supply Chain. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, page 196–209, 2025.

[559] Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent Misalignment: Narrow fine-tuning can produce broadly misaligned LLMs. In *International Conference on Machine Learning (ICML)*, 2025.

[560] Megan Ung, Alicia Sun, Samuel J Bell, Bhaktipriya Radharapu, Levent Sagun, and Adina Williams. Chained tuning leads to biased forgetting. *arXiv preprint arXiv:2412.16469*, 2024.

[561] Chen Xiong, Zhiyuan He, Pin-Yu Chen, Ching-Yun Ko, and Tsung-Yi Ho. Steering Externalities: Benign Activation Steering Unintentionally Increases Jailbreak Risk for Large Language Models. *arXiv preprint arXiv:2602.04896*, 2026.

[562] Divyanshu Kumar, Anurakt Kumar, Sahil Agarwal, and Prashanth Harshangi. Fine-tuning, quantization, and llms: Navigating unintended outcomes. *arXiv preprint arXiv:2404.04392*, 2024.

[563] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16801–16819, 2024.

[564] Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, et al. Tamper-resistant safeguards for open-weight llms. *arXiv preprint arXiv:2408.00761*, 2024.

[565] Saad Hossain, Tom Tseng, Punya Syon Pandey, Samanvay Vajpayee, Matthew Kowal, Nayeema Nonta, Samuel Simko, Stephen Casper, Zhijing Jin, Kellin Pelrine, et al. TamperBench: Systematically Stress-Testing LLM Safety Under Fine-Tuning and Tampering. *arXiv preprint arXiv:2602.06911*, 2026.

[566] Xiangyu Qi, Boyi Wei, Nicholas Carlini, Yangsibo Huang, Tinghao Xie, Luxi He, Matthew Jagielski, Milad Nasr, Prateek Mittal, and Peter Henderson. On evaluating the durability of safeguards for open-weight llms. *arXiv preprint arXiv:2412.07097*, 2024.

[567] Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H. Holliday, Bob M. Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailey Schoelkopf, Emanuel

Tewolde, and William S. Zwicker. Position: Social Choice Should Guide AI Alignment in Dealing with Diverse Human Feedback. In *International Conference on Machine Learning (ICML)*, 2024.

[568] Edoardo Debenedetti, Giorgio Severi, Nicholas Carlini, Christopher A. Choquette-Choo, Matthew Jagielski, Milad Nasr, Eric Wallace, and Florian Tramèr. Privacy side channels in machine learning systems. In *USENIX Security Symposium (USENIX SEC)*, 2024.

[569] Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. Deep learning library testing via effective model generation. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, page 788–799, 2020.

[570] Qianyu Guo, Xiaofei Xie, Yi Li, Xiaoyu Zhang, Yang Liu, Xiaohong Li, and Chao Shen. Audee: automated testing for deep learning frameworks. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, page 486–498, 2021.

[571] Zizhuang Deng, Guozhu Meng, Kai Chen, Tong Liu, Lu Xiang, and Chunyang Chen. Differential Testing of Cross Deep Learning Framework APIs: Revealing Inconsistencies and Vulnerabilities. In *USENIX Security Symposium (USENIX Sec)*, pages 7393–7410, 2023.

[572] Rock Stevens, Octavian Suciu, Andrew Ruef, Sanghyun Hong, Michael Hicks, and Tudor Dumitraş. Summoning demons: The pursuit of exploitable bugs in machine learning. *arXiv preprint arXiv:1701.04739*, 2017.

[573] Google. Google's Secure AI Framework (SAIF). https://safety.google/intl/en_in/safety/saif/, 2025. Accessed: 2026-05-02.

[574] Anthropic. Anthropic's frontier safety roadmap. https://www.anthropic.com/responsible-scaling-policy/roadmap, 2025. Accessed: 2026-05-02.

[575] OpenAI. Openai's preparedness framework. https://openai.com/index/updating-our-preparedness-framework/, 2025. Accessed: 2026-05-02.

[576] Liwei Song and Prateek Mittal. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *USENIX Security Symposium (USENIX SEC)*, pages 2615–2632, 2021.

[577] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks. In *International Conference on Learning Representations (ICLR)*, 2025.

[578] Andrew Jesson, Nicolas Beltran-Velez, Quentin Chu, Sweta Karlekar, Jannik Kossen, Yarin Gal, John P. Cunningham, and David Blei. Estimating the hallucination rate of generative AI. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[579] Legal Dive. Lawyer cites fake cases generated by ChatGPT in legal brief, may 2023.

[580] Sumukh K Aithal, Pratyush Maini, Zachary Chase Lipton, and J Zico Kolter. Understanding Hallucinations in Diffusion Models through Mode Interpolation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[581] Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo {Multi-Turn}{LLM} jailbreak attack. In *USENIX Security Symposium (USENIX SEC)*, pages 2421–2440, 2025.

[582] Xianjun Yang, Liqiang Xiao, Shiyang Li, Faisal Ladhak, Hyokun Yun, Linda Ruth Petzold, Yi Xu, and William Yang Wang. Many-Turn Jailbreaking. *arXiv preprint*, 2025.

[583] Cem Anil, Esin DURMUS, Nina Rimsky, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel J Ford, Francesco Mosconi, Rajashree Agrawal, Rylan Schaeffer, Naomi Bashkansky, Samuel Svenningsen, Mike Lambert, Ansh Radhakrishnan, Carson Denison, Evan J Hubinger, Yuntao Bai, Trenton Bricken, Timothy Maxwell, Nicholas Schiefer, James Sully, Alex Tamkin, Tamera Lanham, Karina Nguyen, Tomasz Korbak, Jared Kaplan, Deep Ganguli, Samuel R. Bowman, Ethan Perez, Roger Baker Grosse, and David Duvenaud. Many-shot Jailbreaking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[584] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 61065–61105, 2024.

[585] Xiaogeng Liu, Peiran Li, G. Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. AutoDAN-Turbo: A Lifelong Agent for Strategy Self-Exploration to Jailbreak LLMs. In *International Conference on Learning Representations (ICLR)*, 2025.

[586] Aiwei Liu, Qiang Sheng, and Xuming Hu. Preventing and Detecting Misinformation Generated by Large Language Models. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, page 3001–3004, 2024.

[587] Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, William Wang, Juan Pino, and Kalika Bali. On the Risk of Misinformation Pollution with Large Language Models. In *Findings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 1389–1403, 2023.

[588] Canyu Chen and Kai Shu. Combating misinformation in the age of llms: Opportunities and challenges. *AI Magazine*, 45(3):354–368, 2024.

[589] Eugene Bagdasaryan and Vitaly Shmatikov. Spinning language models: Risks of propaganda-as-a-service and countermeasures. In *IEEE Symposium on Security and Privacy (S&P)*, pages 769–786, 2022.

[590] Anthropic. Disrupting the First Reported AI-Orchestrated Cyber Espionage Campaign. https://www.anthropic.com/news/disrupting-AI-espionage, 11 2025. News article.

[591] Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI Agents' Real-World Cybersecurity Capabilities at Scale. In *International Conference on Learning Representations (ICLR)*, 2026.

[592] Andy K Zhang, Joey Ji, Celeste Menders, Riya Dulepet, Thomas Qin, Ron Yifeng Wang, Junrong Wu, Kyleen Liao, Jiliang Li, Jinghan Hu, Sara Hong, Nardos Demilew, Shivatmica Murgai, Jason Khiem Tran, Nishka Kacheria, Ethan Jun shen Ho, Denis Liu, Lauren McLane, Olivia Beyer Bruvik, Dai-Rong Han, Seungwoo Kim, Akhil Vyas, Cuiyuanxiu Chen, Ryan Li, Weiran Xu, Jonathan Z Ye, Prerit Choudhary, Siddharth M. Bhatia, Vikram Sivashankar, Yuxuan Bao, Dawn Song, Dan Boneh, Daniel E. Ho, and Percy Liang. BountyBench: Dollar Impact of AI Agent Attackers and Defenders on Real-World Cybersecurity Systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[593] Saskia Laura Schröer, Giovanni Apruzzese, Soheil Human, Pavel Laskov, Hyrum S. Anderson, Edward W. N. Bernroider, Aurore Fass, Ben Nassi, Vera Rimmer, Fabio Roli, Samer Salam, Chi En Ashley Shen, Ali Sunyaev, Tim Wadhwa-Brown, Isabel Wagner, and Gang Wang. SoK: On the Offensive Potential of AI. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 247–280, 2025.

[594] Zhi-Yi Chin, Chieh Ming Jiang, Ching-Chun Huang, Pin-Yu Chen, and Wei-Chen Chiu. Prompting4Debugging: Red-Teaming Text-to-Image Diffusion Models by Finding Problematic Prompts. In *International Conference on Machine Learning (ICML)*, pages 8468–8486, 2024.

[595] Javier Rando, Daniel Paleka, David Lindner, Lennart Heim, and Florian Tramèr. Red-teaming the stable diffusion safety filter. In *ML Safety Workshop, NeurIPS*, 2022.

[596] Ariel Shapira. People are increasingly turning to ChatGPT for affordable therapy—but licensed therapists say there are dangers many aren't considering. *Fortune*, June 2025. Accessed: 2026-02-08.

[597] Tanya Lewis. Why ChatGPT Shouldn't Be Your Therapist. *Scientific American*, August 2025. Accessed: 2026-02-08.

[598] Megan R. Smith. What My Daughter Told ChatGPT Before She Took Her Life. *The New York Times*, August 2025. Opinion.

[599] Forbes Advisor Editorial Staff. Turning To AI Chatbots For Therapy? Why It Might Be More Dangerous Than You Think. *Forbes Advisor*, September 2025. Accessed: 2026-02-08.

[600] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:9460–9471, 2022.

[601] Anthropic. Claude constitution. `https://www.anthropic.com/constitution`, 2025. Accessed: 2026-05-02.

[602] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Annual ACM Symposium on Theory of Computing (STOC)*, page 218–229, 1987.

Appendices

A Chapter 4: Combining ML Defenses without Conflicts

A.1 Formal Analysis of Def\Con

A defense **Def** is defined as a tuple $\texttt{Def} := (Stg, Ch, Rk, Pt)$, where:

- Stg is the stage where Def is applied
- Ch is the type of changes
- Rk is the risk it uses
- Pt is the protection scope of Def, and $\emptyset \notin Pt$.

Let $\mathcal{D}ef$ be the set of all such defenses. The input set for Def\Con is defined as: $X := \{(\texttt{Def}_1, \texttt{Def}_2) | \texttt{Def}_1 \neq \texttt{Def}_2, \ and \ \texttt{Def}_1, \texttt{Def}_2 \in \mathcal{D}ef\}$. Def\Con can be defined as a function: $f : X \to \{0, 1\}$ where $f(\texttt{Def}_1, \texttt{Def}_2) = 0$ indicates conflict and $f(\texttt{Def}_1, \texttt{Def}_2) = 1$ indicates alignment. Given the above formal model, I discuss consistency, soundness, and completeness.

Consistency means that Def\Con never produces contradictory outputs for the same input.

Claim 1. *There is no pair $(\texttt{Def}_1, \texttt{Def}_2)$ for which Def\Con simultaneously classifies both conflict and alignment, i.e., Def\Con is consistent.*

Proof. According to the definition of f, there are two conflict cases:

- (c.1) If $Stg_1 = Stg_2$ and $Ch_1 = Global$ and $Ch_2 = Global$, then $f(\texttt{Def}_1, \texttt{Def}_2) = 0$.
- (c.2) If $Stg_1 \neq Stg_2$ and $Rk_1 \in P_2$, then $f(\texttt{Def}_1, \texttt{Def}_2) = 0$.

For alignment case: (a.1) If none of the conflict conditions hold, then $f(\texttt{Def}_1, \texttt{Def}_2) = 1$.

These conditions partition the input space X into disjoint sets: conflict or alignment, with no overlap. Assume that there exists X_i such that $f(X_i) = 0$ and $f(X_i) = 1$. Since $f(X_i) = 1$, according to (a.1), (c.1) and (c.2) do not hold, leading to a contradiction. Therefore, Def\Con never produces contradictory classifications for the same input and is consistent. □

Soundness ensures that if Def\Con predicts a combination as effective (aligned), it should indeed be effective in practice.

This assumption is supported by my empirical evaluation, where Def\Con achieves:

- 90% accuracy on previously studied defense combinations,
- 81% accuracy on novel, unexplored combinations.

The low rates of false positives and false negatives in these evaluations suggest that Def\Con can reliably predict combination effectiveness, providing practical evidence for its soundness. A perfectly sound technique would rely on comprehensive empirical evaluation, which I aim to avoid by proposing an easy-to-use (approximate) technique, Def\Con. Therefore, the soundness of Def\Con is conditional on the assumption that these rules perfectly represent all conflict and alignment scenarios: (c.1) (c.2) and (a.1). However, since real-world defense interactions can be complex, involving subtle dependencies and emergent behaviors, Def\Con may not fully capture reasons for conflict or alignment. Hence, Def\Con is not perfectly sound and is likely to incur some errors.

Completeness suggests that Def\Con can classify *every possible defense pair* correctly as either conflict or alignment. Similar to soundness, completeness is conditional on the assumption that the classification rules fully capture all conflict and alignment cases. As a partial theoretical guarantee, I can prove that Def\Con can classify every defense pair in its input domain, i.e., it produces a classification for every pair without leaving any case unclassified. Assuming that the input set X includes all possible distinct defense pairs, Def\Con partially satisfies completeness by design. Formally,

Claim 2. *For every pair $(\texttt{Def}_1, \texttt{Def}_2) \in X$, Def\Con's function f produces a classification $f(\texttt{Def}_1, \texttt{Def}_2) \in \{0, 1\}$ that identifies whether the defenses conflict or align, i.e., no conflict or alignment case is left unclassified. Therefore, if X includes all distinct combinations of defenses, Def\Con is partially complete.*

Proof. By construction, Def\Con 's classification rules partition the input space X exhaustively and exclusively by (c.1), (c.2), and (a.1). These rules cover all possible defense pairs in X without overlap or ambiguity, ensuring that every pair is assigned a unique classification; therefore, Def\Con is partially complete on X. □

The assumption that X includes all possible defense combinations is supported by the fact that all surveyed defenses can be represented within the Def\Con framework.

A.2 Formal Analysis for Multi-way Combination Algorithm

Algorithm F handles multi-way combinations by iterating through all permutations of defenses and using Def\Con to check for pairwise conflicts:

(m.1) For defenses $\texttt{Def}_i \in \mathcal{D}ef, i = 1, ..., n$, $F(\texttt{Def}_1, ..., \texttt{Def}_n) = 0$ iff $\exists \texttt{Def}_i, \texttt{Def}_j \in \texttt{Def}$ and $f(\texttt{Def}_i, \texttt{Def}_j) = 0$.

Consistency. According to the consistency of f, assume that there exists $\texttt{Def}_i, i = 1, ..., n$, s.t. $F(\texttt{Def}_1, ..., \texttt{Def}_n) = 0$ and $F(\texttt{Def}_1, ..., \texttt{Def}_n) = 1$. Since $F(\texttt{Def}_1, ..., \texttt{Def}_n) = 1$, $\forall \texttt{Def}_i, \texttt{Def}_j$, we have $f(\texttt{Def}_\mathbf{i}, \texttt{Def}_j) = 1$, leading to a contradiction. Therefore, F is also consistent.

Soundness and Completeness. Under the assumption that f is sound and complete, and considering that we take all possible combinations of defenses (as there is no limit on the input defense set of F), the soundness and completeness of F is conditional on the assumption that (m.1) fully captures all conflict and alignment cases.

Soundness. Since f has some errors and is not perfectly sound, multi-way combinations can indeed introduce conflicts, which will result in false positives or negatives. Hence, the soundness of multi-way Def\Con requires comprehensive evaluation similar to pairwise evaluation. In the evaluation, I only show that effective multi-way combinations (previously unexplored) are possible. A comprehensive evaluation to check for soundness is left as future work.

B Chapter 7: Software-based ML Property Attestation

B.1 Selecting Window Size

Ideally, I want to train f_{att} to distinguish between p_{req} and $!p_{req}$. I compute the AUC scores under the FAR-TAR/FRR-TRR curve when f_{att} is trained to distinguish between p_{req} and $!p_{req}$. I present this in Table B.1 under window size "0". I observe that f_{att} has a high AUC for the edge ratios compared to the middle ratios. Hence, f_{att} is not effective for all p_{req} values.

To this end, I relax the requirement to distinguish between p_{req} and $!p_{req}$. Instead, I evaluate if increasing the window size improves the success of f_{att} in distinguishing between {p_{req}-1, p_{req}, p_{req}+1} and the remaining ratios (!{p_{req}-1, p_{req}, p_{req}+1}). This is indicated in Table B.1 under window size "$\pm$1".

I set a threshold of 0.85 for AUC scores where a score $>= 0.85$ is considered good and marked in green and red otherwise. For ARXIV and BONEAGE, I observe that the window size of $\pm$1 is better than a window size of 0 (visually indicated by more green cells). Hence, for

Table B.1: **Identifying the best window size (w)** using AUC scores to maximize attestation success. red → AUC score <0.85; green → AUC score >=0.85. ±1 → the window {p_{req}-1, p_{req}, p_{req}+1} but only two of those ratios are combined for the edge values. Performance increases when using a window size of ±1 instead of 0. ARXIV and BONEAGE do well across different p_{req} values, but CENSUS-S and CENSUS-R have low effectiveness for certain p_{req} values.

CENSUS-R

Window size	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.00
0	1.00	0.84	0.70	0.81	0.67	0.57	0.67	0.76	0.83	0.92	1.00
±1	0.88	0.89	0.95	0.71	0.74	0.65	0.63	0.83	0.96	0.83	0.75

CENSUS-S

Window size	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.00
0	1.00	0.95	0.74	0.83	0.64	0.55	0.75	0.53	0.72	0.77	1.00
±1	0.93	0.81	0.91	0.79	0.68	0.65	0.63	0.67	0.88	0.89	0.85

BONEAGE

Window size	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0	0.97	0.84	0.88	0.80	0.68	0.80	0.97
±1	0.99	0.99	0.92	0.86	0.87	0.99	0.95

ARXIV

Window size	9	10	11	12	13	14	15	16	17
0	0.99	0.89	0.88	0.77	0.93	0.98	0.99	1.00	1.00
±1	1.00	1.00	0.92	0.96	0.93	0.99	0.99	1.00	1.00

these two datasets, I choose a window size of ± 1 due to better effectiveness. I attribute this to better distinguishability between models trained on datasets with different property values (as shown in [12]).

For CENSUS-R and CENSUS-S, I note that the AUC scores are not high for most ratios in the middle whereas the edge ratios ("0.00" and "1.00") have a perfect score. This suggests that for some p_{req} values in the middle, the cost for attestation is higher than the edge p_{req} values. I attribute this to the lower distinguishability for different p_{req} values (as shown in [12]).

$\mathcal{V}er$ can do a similar analysis beforehand by evaluating on $\mathcal{D}_{te}^{\mathcal{V}er}$ to see the potential cost of attestation for different datasets and identify the window sizes for specific p_{req} values that could improve attestation effectiveness.

B.2 Effectiveness of Robust f_{att}

In the earlier discussion, I highlighted the use of AUC scores for FAR-TAR curves to demonstrate that the performance on clean $\mathcal{D}_{\mathsf{P}}^{vrf}$ remains comparable regardless of whether adversarial training is applied to achieve a robust f_{att}. To provide a more comprehensive assessment of the effectiveness of the robust f_{att}, I present additional metrics in Table B.2, including TRR@5% FRR, TAR@5% FAR, and EER.

Table B.2: **Robust f_{att} effectiveness using TAR, TRR with 5% thresholds for FAR, FRR along with EER across different p_{req} windows on $\mathcal{D}_{\mathsf{P}}^{vrf}$:** The p_{req} value within the window is indicated in **bold**. We observe a similar trend as before: edge p_{req} values have higher effectiveness than middle p_{req} values.

ARXIV			
p_{req} Range	TAR	TRR	EER
{**9**, 10}	0.94	0.94	0.06
{9, **10**, 11}	0.99	0.99	0.02
{10, **11**, 12}	0.20	0.83	0.16
{11, **12**, 13}	0.37	0.59	0.29
{12, **13**, 14}	0.72	0.83	0.11
{13, **14**, 15}	0.86	0.85	0.10
{14, **15**, 16}	0.66	0.86	0.12
{15, **16**, 17}	1.00	1.00	0.00
{16, **17**}	1.00	1.00	0.00

BONEAGE			
p_{req} Range	TAR	TRR	EER
{**0.20**, 0.30}	0.97	0.97	0.05
{0.20, **0.30**, 0.40}	0.99	1.00	0.02
{0.30, **0.40**, 0.50}	0.81	0.87	0.10
{0.40, **0.50**, 0.60}	0.43	0.59	0.23
{0.50, **0.60**, 0.70}	0.26	0.72	0.25
{0.60, **0.70**, 0.80}	0.97	0.98	0.04
{0.70, **0.80**}	0.93	0.94	0.05

CENSUS-S			
p_{req} Range	TAR	TRR	EER
{**0.00**}	1.00	1.00	0.00
{0.00, **0.10**, 0.20}	0.45	0.56	0.21
{0.10, **0.20**, 0.30}	0.57	0.66	0.17
{0.20, **0.30**, 0.40}	0.20	0.44	0.26
{0.30, **0.40**, 0.50}	0.11	0.29	0.38
{0.40, **0.50**, 0.60}	0.14	0.18	0.39
{0.50, **0.60**, 0.70}	0.16	0.19	0.38
{0.60, **0.70**, 0.80}	0.16	0.22	0.34
{0.70, **0.80**, 0.90}	0.55	0.58	0.19
{0.80, **0.90**, 1.00}	0.62	0.61	0.19
{**1.00**}	1.00	1.00	0.00

CENSUS-R			
p_{req} Range	TAR	TRR	EER
{**0.00**}	1.00	1.00	0.00
{0.00, **0.10**, 0.20}	0.06	0.51	0.36
{0.10, **0.20**, 0.30}	0.66	0.86	0.10
{0.20, **0.30**, 0.40}	0.22	0.49	0.29
{0.30, **0.40**, 0.50}	0.10	0.23	0.39
{0.40, **0.50**, 0.60}	0.12	0.13	0.41
{0.50, **0.60**, 0.70}	0.10	0.25	0.38
{0.60, **0.70**, 0.80}	0.10	0.39	0.29
{0.70, **0.80**, 0.90}	0.66	0.82	0.12
{0.80, **0.90**, 1.00}	0.49	0.59	0.22
{**1.00**}	1.00	1.00	0.00

Similar to the outcomes shown in Table 2 for the non-robust f_{att}'s effectiveness, I observe elevated TRR and TAR values for edge p_{req} values, but diminished performance for the middle

p_{req} values. This trend can be attributed to differences in distinguishability, a notion underscored by Suri and Evans [12]. Additionally, I find that some EER values for specific p_{req} values outperform both FAR and FRR, suggesting the potential for a superior threshold beyond the standard 5%.

However, it is important to note that despite this improvement, the ML-based mechanism utilizing a robust f_{att} remains ineffective for certain p_{req} values. This implies that, despite its robustness, relying solely on the ML-based mechanism is inadequate.

B.3 Impact of Adversarial Fine-Tuning Attack on $\mathcal{M}$'s Accuracy

I examined the capabilities of $\mathcal{P}rv_{mal}$ who employs an adversarial fine-tuning attack strategy. This involves introducing adversarial noise to the parameters of the first layer, followed by the fine-tuning of subsequent layers. Consequently, $\mathcal{P}rv_{mal}$ is able to fool $\mathcal{V}er$'s attestation process by exploiting the adversarially perturbed first layer parameters.

I indicate the impact of perturbing the first layer parameters on the $\mathcal{P}rv_{mal}$'s $\mathcal{M}$ accuracy in Table B.3. I observe that adding noise to the first layer followed by fine-tuning restores the accuracy of the model ($\mathcal{A}_{att}$) close to the original accuracy ($\mathcal{A}_{orig}$) while still being able to evade the attestation. Hence, the attack can be done without a significant decrease in accuracy.

B.4 Details for Cryptographic Mechanism

Protocol Instantiation. Given the proof objectives, the goal is to find a concrete cryptographic protocol variant to instantiate property attestation. To this end, I need the following: (1) primitives for ML training, (2) security against $\mathcal{P}rv_{mal}$, and (3) an efficient protocol instantiation that allows one to use the cryptographic mechanism in a real setting. I rule out TEEs because of their susceptibility to side-channel attacks, hence violating (2). Because of their impracticality to deploy in real-world sized models, thus violating (3), I also rule out Homomorphic Encryption (HE) [514]. As of now, there are efficient ZKPs for verifiable inference [498], but not for backpropagation during ML training, violating (1) and ruling out ZKPs for this instantiation. As state-of-the-art works in PPML based on MPC such as CrypTen [476] satisfy all three required properties for the cryptographic mechanism, I choose MPC as the instantiation.

Property Attestation as a Cryptographic Protocol. Assuming a malicious $\mathcal{P}rv$ and semi-honest $\mathcal{V}er$, I construct a cryptographic protocol based on MPC in the outsourcing setting with two non-colluding semi-honest servers $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$. The protocol has the following steps:

Table B.3: **Average accuracy across 1000** shadow models w/ and w/o malicious $\mathcal{P}rv$ modifying the first layer parameters followed by fine-tuning. We refer to them as $\mathcal{A}_{orig}$ and $\mathcal{A}_{att}$, respectively. green indicates $\mathcal{A}_{orig} >> \mathcal{A}_{att}$, orange indicates $\mathcal{A}_{orig} > \mathcal{A}_{att}$ (<2%) and red indicates $\mathcal{A}_{orig} \sim \mathcal{A}_{att}$. Across most p_{req} values, malicious $\mathcal{P}rv$ can restore $\mathcal{A}_{att}$ close to $\mathcal{A}_{orig}$.

ARXIV

p_{req} **Range**	Shadow Model Accuracy	
	$\mathcal{A}_{orig}$	$\mathcal{A}_{att}$
{**9**, 10}	0.72 ± 0.00	0.70 ± 0.00
{9, **10**, 11}	0.72 ± 0.00	0.70 ± 0.00
{10, **11**, 12}	0.72 ± 0.00	0.71 ± 0.00
{11, **12**, 13}	0.72 ± 0.00	0.71 ± 0.00
{12, **13**, 14}	0.72 ± 0.00	0.71 ± 0.00
{13, **14**, 15}	0.72 ± 0.00	0.71 ± 0.00
{14, **15**, 16}	0.72 ± 0.00	0.72 ± 0.00
{15, **16**, 17}	0.72 ± 0.00	0.72 ± 0.00
{16, **17**}	0.72 ± 0.00	0.72 ± 0.01

BONEAGE

p_{req} **Range**	Shadow Model Accuracy	
	$\mathcal{A}_{orig}$	$\mathcal{A}_{att}$
{**0.20**, 0.30}	0.79 ± 0.02	0.79 ± 0.02
{0.20, **0.30**, 0.40}	0.78 ± 0.02	0.76 ± 0.02
{0.30, **0.40**, 0.50}	0.78 ± 0.02	0.75 ± 0.02
{0.40, **0.50**, 0.60}	0.78 ± 0.02	0.70 ± 0.02
{0.50, **0.60**, 0.70}	0.78 ± 0.02	0.78 ± 0.01
{0.60, **0.70**, 0.80}	0.78 ± 0.02	0.78 ± 0.01
{0.70, **0.80**}	0.78 ± 0.01	0.79 ± 0.01

CENSUS-S

p_{req} **Range**	Shadow Model Accuracy	
	$\mathcal{A}_{orig}$	$\mathcal{A}_{att}$
{**0.00**}	0.77 ± 0.04	0.80 ± 0.01
{0.00, **0.10**, 0.20}	0.79 ± 0.03	0.82 ± 0.01
{0.10, **0.20**, 0.30}	0.79 ± 0.03	0.82 ± 0.01
{0.20, **0.30**, 0.40}	0.80 ± 0.02	0.82 ± 0.01
{0.30, **0.40**, 0.50}	0.81 ± 0.02	0.82 ± 0.01
{0.40, **0.50**, 0.60}	0.82 ± 0.02	0.83 ± 0.01
{0.50, **0.60**, 0.70}	0.82 ± 0.02	0.84 ± 0.01
{0.60, **0.70**, 0.80}	0.82 ± 0.04	0.84 ± 0.01
{0.70, **0.80**, 0.90}	0.83 ± 0.02	0.84 ± 0.01
{0.80, **0.90**, 1.00}	0.85 ± 0.05	0.86 ± 0.01
{**1.00**}	0.87 ± 0.03	0.88 ± 0.01

CENSUS-R

p_{req} **Range**	Shadow Model Accuracy	
	$\mathcal{A}_{orig}$	$\mathcal{A}_{att}$
{**0.00**}	0.87 ± 0.04	0.87 ± 0.01
{0.00, **0.10**, 0.20}	0.86 ± 0.03	0.87 ± 0.01
{0.10, **0.20**, 0.30}	0.86 ± 0.02	0.86 ± 0.01
{0.20, **0.30**, 0.40}	0.85 ± 0.01	0.85 ± 0.01
{0.30, **0.40**, 0.50}	0.84 ± 0.02	0.85 ± 0.01
{0.40, **0.50**, 0.60}	0.84 ± 0.02	0.85 ± 0.01
{0.50, **0.60**, 0.70}	0.83 ± 0.02	0.84 ± 0.01
{0.60, **0.70**, 0.80}	0.82 ± 0.02	0.83 ± 0.01
{0.70, **0.80**, 0.90}	0.81 ± 0.02	0.82 ± 0.01
{0.80, **0.90**, 1.00}	0.81 ± 0.02	0.82 ± 0.01
{**1.00**}	0.81 ± 0.02	0.82 ± 0.01

- ***Initiate input-sharing phase between*** $\mathcal{P}rv$ ***and*** $\mathcal{S}rv_1$, $\mathcal{S}rv_2$: $\mathcal{P}rv$ computes additive secret-shares of $\mathcal{D}_{tr}^{prv}$. Hence, $\mathcal{P}rv$ computes $[\mathcal{D}_{tr}^{\mathcal{P}rv}]^1$, $[\mathcal{D}_{tr}^{\mathcal{P}rv}]^2$ such that $[\mathcal{D}_{tr}^{\mathcal{P}rv}]^1 + [\mathcal{D}_{tr}^{\mathcal{P}rv}]^2 = \mathcal{D}_{\mathcal{P}}^{tr}$ and sends $[\mathcal{D}_{tr}^{\mathcal{P}rv}]^1$ to $\mathcal{S}rv_1$ and $[\mathcal{D}_{tr}^{\mathcal{P}rv}]^2$ to $\mathcal{S}rv_2$.
- $\mathcal{S}rv_1$ ***and*** $\mathcal{S}rv_2$ ***run*** `DistCheck` ***on their input shares of*** $\mathcal{D}_{tr}^{prv}$: Given the input shares of $\mathcal{D}_{tr}^{prv}$, $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ compute `DistCheck` by computing the distributional property of $\mathcal{D}_{tr}^{prv}$ and comparing against p_{req}.
- $\mathcal{S}rv_1$ ***and*** $\mathcal{S}rv_2$ ***securely train on their input shares, which yields*** $\mathcal{M}_{2pc}$***:*** Given the input shares of both $\mathcal{D}_{tr}^{prv}$ and $\mathcal{M}$, both servers jointly run the protocols for secure training as described in [476]. CrypTen has efficient secure protocols for both the forward pass and back propagation. I refer to [476] for the protocol details. I emphasize that $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ use the previously obtained shares of $\mathcal{D}_{tr}^{prv}$ from the input-sharing phase, because they were

used for `DistCheck`. This leaves no room for $\mathcal{P}rv$ to cheat by choosing different shares of another dataset $\mathcal{D}' \neq \mathcal{D}^{tr}_{\mathcal{P}}$ for training.

- $\mathcal{S}rv_1$ ***and*** $\mathcal{S}rv_2$ ***send output shares of*** `DistCheck` ***and*** $\mathcal{M}_{2pc}$ ***to*** $\mathcal{V}er$ ***for reconstruction of plaintext outputs***: $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$ send the output shares $[v]^1$ and $[v]^2$ of `DistCheck` to $\mathcal{V}er$ who now locally reconstructs the output $v = [v]^1 + [v]^2$. Now, $v = 1$ iff `DistCheck` was successful. Then, $\mathcal{V}er$ reconstructs $\mathcal{M}_{2pc}$ by locally adding the output shares from $\mathcal{S}rv_1$ and $\mathcal{S}rv_2$, i.e., $\mathcal{M}_{\text{2PC}} = [\mathcal{M}_{2pc}]^1 + [\mathcal{M}_{2pc}]^2$.
- $\mathcal{V}er$ ***checks if*** `DistCheck` ***succeeded using the output shares.***

Security and Correctness of Cryptographic Mechanism. Since I implement the cryptographic mechanism using CrypTen, I refer to [476] for the detailed security proofs. Assuming CrypTen's protocols satisfy security and correctness, I discuss the security (i.e., preserving the privacy of $\mathcal{P}rv$'s dataset $\mathcal{D}^{prv}_{tr}$) and correctness for cryptographic mechanism. The privacy of $\mathcal{D}^{prv}_{tr}$ naturally follows from the security guarantees of linear secret-sharing [602, 476]. For correctness, I identify two cases:

- **when** $\mathcal{P}rv$ **does not cheat** and correctly creates an input sharing for $\mathcal{D}^{prv}_{tr}$, then, correctness follows from the underlying secret-sharing scheme.
- **when** $\mathcal{P}rv$ **cheats**, then $\mathcal{P}rv$ can
  - simply abort instead of providing a valid sharing to escape attestation, hence the whole attestation fails.
  - create incorrect shares $[\mathcal{D}']^1$ and $[\mathcal{D}']^2$ such that $[\mathcal{D}']^1 + [\mathcal{D}']^2 = \mathcal{D}' \neq \mathcal{D}_{\mathcal{P}}$. However, since the two shares indeed compute $\mathcal{D}'$, the input sharing is performed correctly, just for a different input value. MPC does not protect against choosing the "wrong" input value. Nevertheless, this is not a problem, because if $\mathcal{D}'$ satisfies $\preceq$, we still obtain a valid model with respect to the distributional property.

After the input-sharing phase, $\mathcal{P}rv$ does not participate in the protocol. Hence, there is no further cheating as $\mathcal{S}rv_1$, $\mathcal{S}rv_2$, $\mathcal{V}er$ are semi-honest. Secure computation of `DistCheck` consists only of secure additions and comparisons; hence, the correctness and privacy of `DistCheck` as well as secure training of $\mathcal{M}_{2pc}$ directly follow from the security guarantees of CrypTen [476].

B.5 Hybrid Attestation: FAR vs. Cost on $\mathcal{D}^{vrf}_{\mathbf{p}}$

For hybrid attestation, I show the trade-off between FAR and expected cost for varying $\mathcal{N}_{spchk}$ on $\mathcal{D}^{\mathcal{V}er}_{te}$. $\mathcal{V}er$ uses this to identify the suitable value of $\mathcal{N}_{spchk}$. To get the actual estimate of FAR and expected cost, $\mathcal{V}er$ can use the chosen value of $\mathcal{N}_{spchk}$ in a plot between FAR and expected cost in Figure B.2.

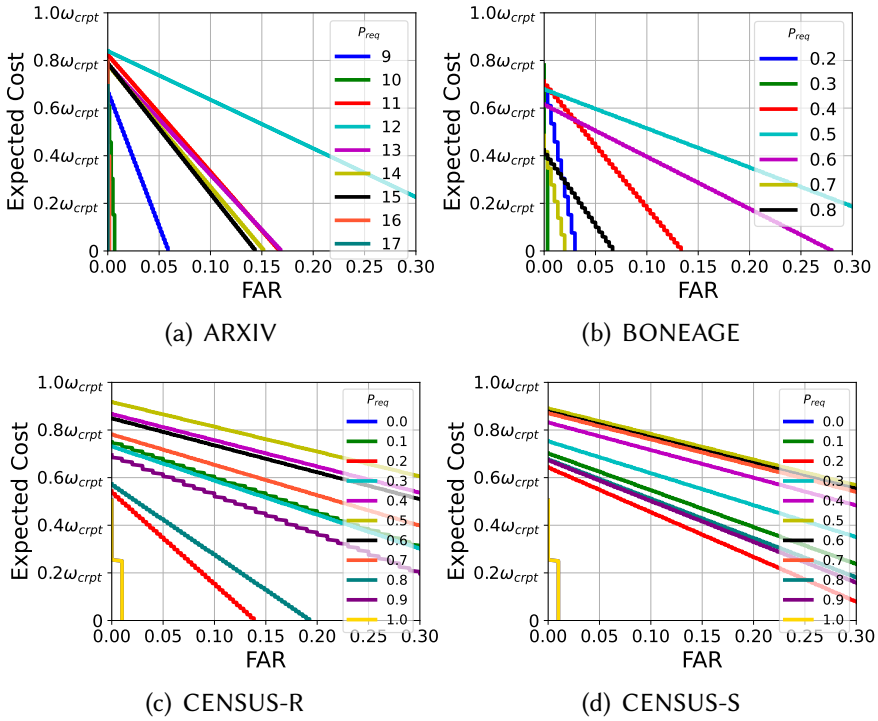


Figure B.2: **Fixed FRR@5%: Expected cost on varying $\mathcal{N}_{spchk}$ on $\mathcal{D}^{vrf}_{\mathbf{P}}$.** $\mathcal{V}er$ can use suitable value of $\mathcal{N}_{spchk}$ identified from $\mathcal{D}^{\mathcal{V}er}_{te}$ to note the corresponding FAR and expected cost.

I still observe that the expected cost is lower for the hybrid mechanism than for the cryptographic mechanism for all accepted $\mathcal{P}rv$s, and the FAR is lower than with no spot-checks (same as the ML-based mechanism).