Assumption Stress-Testing for Machine Learning Security

by

Andre Kassis

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

© Andre Kassis 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:	Neil Zhenqiang Gong Associate Professor, Dept. of Electrical and Computer Engineering Duke University
Supervisor(s):	Urs Hengartner Associate Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal Members:	N. Asokan Professor, David R. Cheriton School of Computer Science University of Waterloo
	Yaoliang Yu Associate Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Otman Basir Professor, Dept. of Electrical and Computer Engineering University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see the 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

This dissertation includes first-authored work that has appeared in peer-reviewed conference proceedings, primarily at the IEEE Symposium on Security and Privacy (S&P) and the Conference on Neural Information Processing Systems (NeurIPS). In all cases, I led the research, including problem formulation, methodology design, implementation, evaluation, and writing. This work was conducted under the supervision of Prof. Urs Hengartner, who provided guidance and feedback throughout. For Chapter 5, Prof. Yaoliang Yu also provided valuable feedback and discussions.

The following list serves as a declaration of the works included in this dissertation. I expand and revise material from the original publications.

Chapter 3

Andre Kassis and Urs Hengartner. 2023. Breaking Security-Critical Voice Authentication. Published in the *Proceedings of the 44th IEEE Symposium on Security and Privacy*.

Chapter 4

Andre Kassis and Urs Hengartner. 2025. *UnMarker*: A Universal Attack on Defensive Image Watermarking. Published in the *Proceedings of the 46th IEEE Symposium on Security and Privacy*.

Chapter 5

Andre Kassis, Urs Hengartner and Yaoliang Yu. 2025. *DiffBreak*: Is Diffusion-Based Purification Robust? Published in the *Proceedings of the 39th Conference on Neural Information Processing Systems*.

Chapter 6

Andre Kassis and Urs Hengartner. 2026. *KEYSHIFT*: Key-Induced Boundary Shifts for Adversarial Robustness. *Under submission*.

Abstract

Security is guided by the principle that a system is only as strong as its weakest link. Despite the undeniable benefits brought by the growing integration of machine learning (*ML*) into nearly all aspects of modern life, decades of research and real-world deployment have established a broad consensus that systems oblivious to malicious behavior are fundamentally unsafe. In practice, ill-intended users can abuse access to powerful generative models to create harmful deepfakes capable of facilitating fraud and impersonation, or instigating societal chaos. At the same time, the rapid adoption of *ML* in high-stakes settings such as online moderation, media authenticity verification, and biometric authentication introduces the risk of these mechanisms being manipulated or bypassed through carefully crafted inputs with potentially catastrophic consequences, as repeatedly demonstrated in practice.

This thesis approaches these threats from an assumption-driven perspective, arguing that critical failures in *ML* security do not arise from isolated weaknesses of individual models, but from flawed or insufficiently scrutinized assumptions underlying broader robustness concepts. To study this problem, the thesis advances the perspective of *assumption stress-testing*, a framework for systematically analyzing the assumptions on which security mechanisms rely and constructing targeted approaches that subject them to worst-case behavior. Across four projects spanning multiple *ML* security domains, the thesis demonstrates how violating these assumptions can make the underlying concepts themselves inapplicable, causing entire classes of mechanisms built upon them to fail regardless of how they are instantiated. Specifically, the thesis investigates the robustness of voice authentication and image watermarking technologies designed to support media authenticity, revisits the reliability of a state-of-the-art defense paradigm against adversarial examples, and explores the constructive use of assumption-aware analysis to devise systems whose robustness remains meaningful under rigorous evaluation.

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Urs Hengartner, whose guidance has shaped every page of this thesis. Over the years, you consistently offered insightful and constructive feedback that sharpened my thinking and elevated my work, and did so with a patience and generosity I will always be grateful for. Your mentorship taught me not only how to conduct rigorous research, but how to be a researcher. This work would not have been possible without you.

I am also deeply grateful to my committee members: Prof. N. Asokan, Prof. Yaoliang Yu, Prof. Otman Basir, and Prof. Neil Gong. I know how demanding your schedules are, and I sincerely appreciate the time you devoted to reading this thesis and your willingness to serve on my committee. Your thoughtful questions and comments have made this work unquestionably better. I owe special thanks to Asokan and Yaoliang, whose support went well beyond their roles on my committee. Throughout my journey, you offered advice, encouragement, and help whenever I needed it, and your doors were always open. I truly appreciate everything you have done for me.

I would like to thank Dr. Elie Khoury, Senior Vice President at Pindrop Security, from whom I learned a great deal during my time at Pindrop. Elie, you have remained both a mentor and a friend ever since, extending guidance and help as I navigate the next chapter of my career. I am very thankful for that.

To my family, and especially my parents, Boulos and Karawan Kassis: you set me on this path long before I knew where it would lead. From my earliest years, you instilled in me a curiosity, a love of learning, and a drive to solve problems, and you have supported me through everything—even the hardest of times. I am eternally grateful for all you have done. Without you, my life would look entirely different, and none of this would have been possible. I also thank my siblings, Hala and Martin, and my cousin Michael Kassis, who has been an anchor I could rely on since the day I arrived in Canada. I truly appreciate all your help.

Finally, and most importantly, I thank my wife, Saira Kassis, the love of my life. Your patience, love, and devotion throughout this journey have been beyond measure. No one else could have endured the stress and hardship these years brought—and yet you never once left my side. You cared for me and went to great lengths to make sure I could stay focused and pursue my goals, and without you, I would likely have given up long ago. This thesis is yours as much as it is mine. I am so glad to have closed this chapter, and I cannot wait to embark on the next journey together.

Dedication

To my wife, Saira, my parents, Boulos and Karawan, and the memory of my grandmother, Nejme Kassis.

Table of Contents

Examining Committee Membership ii

Author's Declaration iii

Statement of Contributions iv

Abstract v

Acknowledgements vi

Dedication vii

List of Figures xv

List of Tables xix

1 Introduction 1

1.1 Contributions 2

1.2 Overview 4

2 Background 5

2.1 Machine Learning Foundations 5

2.1.1 *ML* Classifiers 5

2.1.2 Generative & Diffusion Models . . . 6
2.2 Content Authenticity Mechanisms . . . 8
2.2.1 Passive Detection . . . 9
2.2.2 Watermarking . . . 9
2.3 Adversarial Attacks and Defenses . . . 11
2.3.1 Adversarial Examples . . . 11
2.3.2 Adversarial Defenses . . . 15

3 Breaking Security-Critical Voice Authentication 17
3.1 Motivation . . . 17
3.2 Background . . . 20
3.2.1 Automatic Speaker Verification (ASV) . . . 20
3.2.2 Spoofing Countermeasures (CMs) . . . 21
3.2.3 Speech-to-text Systems (STTs) . . . 22
3.2.4 VA in Security-Critical Deployments . . . 22
3.3 Threat Model . . . 23
3.3.1 Definitions . . . 23
3.3.2 Attack Channels . . . 23
3.3.3 Attacker's Goals, Knowledge, and Capabilities . . . 23
3.4 Attacking Security-Critical VA Deployments . . . 25
3.4.1 Attack Overview . . . 25
3.4.2 Universal Adversarial Transformations Against CMs . . . 26
3.4.3 Attack Implementation . . . 30
3.5 Experimental Setup . . . 31
3.6 Attacking Designated-App Authentication . . . 34
3.6.1 Results Against CMs . . . 35
3.6.2 Results Against ASVs . . . 37
3.6.3 Results Against STTs . . . 38

3.6.4 Results Against the Combined Defense . . . 38
3.7 Over-Telephony-Network Attacks . . . 40
3.8 Fooling a Human Judge . . . 40
3.9 Discussion . . . 43
3.10 Related Work . . . 45
3.11 Conclusion . . . 47

4 UnMarker: A Universal Attack on Defensive Image Watermarking 48
4.1 Motivation . . . 48
4.2 Background & Related Work . . . 51
4.2.1 *Semantic* vs. *Non-Semantic* Watermarking . . . 51
4.2.2 Attacks Against Defensive Watermarking . . . 52
4.3 Threat Model . . . 53
4.4 Design Philosophy . . . 54
4.4.1 Uncovering The Universal Carrier . . . 54
4.5 UnMarker . . . 59
4.5.1 Attacking Non-Semantic Watermarks . . . 60
4.5.2 Attacking Semantic Watermarks . . . 62
4.6 Experiments . . . 69
4.6.1 Setup . . . 69
4.6.2 Evaluations . . . 71
4.7 Discussion . . . 78
4.7.1 Practicality . . . 78
4.7.2 Mitigations . . . 79
4.7.3 Outlook . . . 80
4.7.4 Limitations & Future Work . . . 82
4.7.5 Ethical Considerations . . . 84
4.8 Conclusion . . . 84

5 DiffBreak: Is Diffusion-Based Purification Robust? 85
5.1 Motivation . . . 85
5.2 Background & Related Work . . . 89
5.3 Revisiting Diffusion-Based Purification . . . 91
5.3.1 Why *DBP* Fails: Theoretical Vulnerability . . . 91
5.3.2 Precise *DBP* Gradients with **DiffGrad** . . . 95
5.3.3 On The Issues in *DBP*'s Evaluation Protocols. . . . 97
5.4 Experiments . . . 98
5.4.1 Reassessing One-Shot *DBP* Robustness with Accurate Gradients . . . 98
5.4.2 *DBP* Under Realistic Protocols . . . 99
5.5 Defeating Increased Stochasticity . . . 103
5.5.1 *DBP* Against Low-Frequency *AE*s . . . 105
5.6 Potential Countermeasures . . . 106
5.7 Conclusion . . . 107

6 KEYSHIFT: Key-Induced Boundary Shifts for Adversarial Robustness 109
6.1 Motivation . . . 109
6.2 Preliminaries & Related Work . . . 111
6.2.1 Threat Model & Overview . . . 111
6.2.2 Related Work . . . 116
6.3 KEYSHIFT . . . 117
6.3.1 Guiding Objectives . . . 117
6.3.2 System Design . . . 118
6.4 Evaluations on Common Benchmarks . . . 122
6.4.1 Setup . . . 123
6.4.2 Results . . . 124
6.5 Evaluations against Advanced Adaptive Attacks . . . 126
6.5.1 Black-box Attacks . . . 127

6.5.2 Extraction Attacks . . . 128
6.6 Discussion & Limitations . . . 136
6.7 Conclusion . . . 137

7 Conclusion & Future Directions **138**
7.1 Summary of Contributions . . . 139
7.2 Future Directions . . . 141

Letter of Copyright Permission **144**

References **146**

APPENDICES **176**

A Breaking Security-Critical Voice Authentication **177**
A.1 ASV Models . . . 177
A.2 CM Calibration for The Over-Telephony Setting . . . 177
A.3 Shattering the Myth of Transferability . . . 179
A.4 Results for Individual Transformations . . . 180
A.5 Content Preservation . . . 182

B UnMarker: A Universal Attack on Defensive Image Watermarking **184**
B.1 *UnMarker*'s Hyperparameters . . . 184
B.2 Watermarking Schemes' Configuration . . . 186
B.3 Baseline Results against Image Manipulations . . . 186
B.4 Outputs of *UnMarker* vs. The *SOTA* . . . 188

C DiffBreak: Is Diffusion-Based Purification Robust? **190**
C.1 Equivalence of **DDPM** and **VP-SDE** . . . 190
C.2 Details on Our **DiffGrad** Gradient Module . . . 191

C.2.1 The Challenges of Applying Checkpointing to *DBP* and How **DiffGrad** Tackles Them . . . 191
C.2.2 Demonstrating The Effects of Incorrect Backpropagation . . . 195
C.2.3 Pseudo-Code for Our Memory-Efficient Gradient-Enabled Purification with **DiffGrad** . . . 199
C.2.4 Verifying The Correctness of **DiffGrad** . . . 203
C.3 Additional Details on Systems & Models . . . 204
C.4 One-Shot *DBP* Baseline Comparisons Against **DiffGrad**'s Accurate Gradients 204
C.5 Evaluations with Liu et al. [162]'s Fixed *AutoAttack* . . . 206
C.6 Ablation Study . . . 207
C.7 Details on Our Low-Frequency (*LF*) Adversarial Optimization Strategy . . . 210
C.7.1 Attack Hyperparameters . . . 210
C.7.2 Pseudo-Code . . . 210
C.8 Example Attack Images . . . 213
C.8.1 Attack Samples Generated Against ***CIFAR-10***'s WideResNet-70-16 with *GDMP* Purification . . . 214
C.8.2 Attack Samples Generated Against ***CIFAR-10***'s WideResNet-28-10 with *GDMP* Purification . . . 214
C.8.3 Attack Samples Generated Against ***CIFAR-10***'s WideResNet-70-16 with *DiffPure* Purification . . . 215
C.8.4 Attack Samples Generated Against ***CIFAR-10***'s WideResNet-28-10 with *DiffPure* Purification . . . 216
C.8.5 Attack Samples Generated Against ***ImageNet***'s DeiT-S with *GDMP* Purification . . . 217
C.8.6 Attack Samples Generated Against ***ImageNet***'s WideResNet-50-2 with *GDMP* Purification . . . 219
C.8.7 Attack Samples Generated Against ***ImageNet***'s ResNet-50 with *GDMP* Purification . . . 220
C.8.8 Attack Samples Generated Against ***ImageNet***'s DeiT-S with *DiffPure* Purification . . . 221

C.8.9 Attack Samples Generated Against ***ImageNet***'s WideResNet-50-2 with *DiffPure* Purification 223
C.8.10 Attack Samples Generated Against ***ImageNet***'s ResNet-50 with *DiffPure* Purification 225
C.8.11 Quality Comparison with *StAdv* 225

D KEYSHIFT: Key-Induced Boundary Shifts for Adversarial Robustness 228
D.1 Design & Training 228
D.1.1 ConvolveCond 228
D.1.2 U-Net Integration 229
D.1.3 Preprocessor Training 230
D.2 Experimental Setup Details 235
D.2.1 Key Space Construction 235
D.2.2 Preprocessor Configuration 235
D.2.3 Training Configuration 236
D.2.4 Adversarial Attack Configuration 238
D.3 Blackbox Attacks 239
D.4 Complexity Discussion 242

List of Figures

3.1 Attack Overview. 26

3.2 *Results for Tasks 1 and 2.* 42

4.1 Spectral analysis of two images. Both depict worldly objects of high consistency. Thus, the ***collective*** spectral magnitudes are distributed similarly as the low frequencies (center) corresponding to gradual variations are always far more dominant. Phases determine the spatial arrangement of the pixel value shifts that contribute to these magnitudes to shape the content, making them extremely different. 56

4.2 Images watermarked by *StableSignature*— *non-semantic* (top), and *StegaStamp*— *semantic* (bottom). The rightmost figures display the differences between the original and watermarked images that correspond to the changes encoding the watermarks. *StableSignature*'s modifications are restricted to existing (high-frequency) edges such as wrinkles, hair, mustache, and intersections of multiple components. *StegaStamp*'s watermark is distributed across the image. The magnified area shows how it manipulates the consistency (texture), injecting gradual (low-frequency) changes that manifest as wrinkles at this location. 57

4.3 Operation of known filters. 63

5.1 **DiffGrad**. $\boldsymbol{x}$ is given to the iterative attack algorithm $\mathcal{A}_\theta$ to generate $\boldsymbol{x}_{adv}$. At each iteration, $\boldsymbol{x}_{adv}$ is propagated through *DBP*, yielding $\hat{\boldsymbol{x}}_{adv}(0)$ that is given to $\mathcal{M}$ while storing each intermediate $\hat{\boldsymbol{x}}_{adv}(t)$ (bottom replicas) from *DBP*'s reverse pass (see §5.2) but without saving any graph dependencies. Backpropagation uses the stored samples: Starting from $t=-dt$ ($dt<0$—see §5.2), each $\hat{\boldsymbol{x}}_{adv}(t)$ is used to recompute $\hat{\boldsymbol{x}}_{adv}(t+dt)$, retrieving the required dependencies. Then, we recursively obtain the gradient w.r.t. $\hat{\boldsymbol{x}}_{adv}(t)$ from the gradients w.r.t. $\hat{\boldsymbol{x}}_{adv}(t+dt)$ using this recovered sub-graph (see §5.3.2). Finally, gradients are backpropagated in a standard manner from $\boldsymbol{x}_{adv}(t^*)$ to $\mathcal{A}_\theta$ to update $\boldsymbol{x}_{adv}$. 87

6.1 **KEYSHIFT deployment.** A **pretrained** classifier $\mathcal{M}$ is protected by a **keyed** preprocessor $P(\boldsymbol{x},\boldsymbol{z})$. For non-secret keys $\boldsymbol{z}\neq\boldsymbol{z}^\star$ (public mode), $P(\boldsymbol{x},\boldsymbol{z})\approx\boldsymbol{x}$ and the system behaves as the original classifier, predicting the ground-truth y. Under the secret key $\boldsymbol{z}^\star$ (**private mode**, center), $P(\boldsymbol{x},\boldsymbol{z}^\star)$ generates a sample-dependent transformation that causes $\mathcal{M}$ to predict $\pi(y)$, where π is a permutation defining a decision function known only to the model owner. User inputs (left) are evaluated in private mode and π is inverted to infer the label (users **never** see the permuted $\pi(y)$). Attackers (right) who know the weights and architectures **but not $\boldsymbol{z}^\star$ or π** (Kerckhoffs') may craft attacks locally in **public mode** (i.e., by providing $\boldsymbol{z}\neq\boldsymbol{z}^\star$ during optimization), but absent $\boldsymbol{z}^\star$ they attack mismatched boundaries, causing *AE*s to fail under private-mode evaluation—see Fig. 6.2. Effective attacks thus require estimating the private behavior; under realistic security-critical deployment constraints (e.g., rate limiting, hard-label access), black-box strategies are ineffective, and *KEYSHIFT* is constructed to render extraction of the hidden elements information-limited (see §6.5). 112

6.2 **Non-transfer via mode-dependent space.** An adversarial perturbation $\boldsymbol{\delta}$ is crafted in public mode (left), moving $\boldsymbol{x}$ into the misclassification region $\mathcal{R}_{\text{pub}}$ and flipping the label. Under the owner's configuration (private mode, right), the induced representation and decision region differ; the *same* input maps to a different point in feature space, and the movement fails to cross the misclassification region $\mathcal{R}_{\text{priv}}$. 117

6.3 **Key-space geometry.** Non-secrets collapse to a single behavior with negligible sensitivity to $\boldsymbol{z}$. $\boldsymbol{z}^\star$ activates an isolated function. Searches over $\boldsymbol{z}$ receive no signal outside the secret region. 119

B.1 Outputs of the three removal attacks. The *VAEAttack* over-smooths images, resulting in them losing crucial information and occasionally looking cartoonish. Compared to the *DiffusionAttack*, *UnMarker*'s outputs better resemble the watermarked images. The *DiffusionAttack* also introduces semantic incoherences such as additional teeth in the case of *PTW* and omits identifying details such as freckles on the forehead for *StegaStamp*. *PTW*'s watermarked image itself is of inferior quality with an unnatural patch. The *DiffusionAttack* retains this artifact while *UnMarker* eliminates it, enhancing the image. 189

C.1 Effects of the identified issues in *DBP*'s backpropagation. Each subfigure visualizes a specific error source. 196

C.2 Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*. 214

C.3 Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*. 214

C.4 Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*. 215

C.5 Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*. 215

C.6 Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*. 215

C.7 Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*. 215

C.8 Successful attacks generated with *LF*. Left -original image. Right - *LF*. . . . 216

C.9 Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*. 217

C.10 Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*. 218

C.11 Successful attacks generated with *LF*. Left -original image. Right - *LF*. . . . 219

C.12 Successful attacks generated with *LF*. Left -original image. Right - *LF*. . . . 220

C.13 Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*. 221

C.14 Successful *LF* attacks on inputs for which AA-ℓ_∞ fails. Left - original image. Right - *LF*. 222

C.15 Successful attacks generated by *LF* and AA-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*. 223

C.16 Successful *LF* attacks on inputs for which AA-ℓ_∞ fails. Left - original image. Right - *LF*. 224

C.17 Successful attacks generated with *LF*. Left -original image. Right - *LF*. . . 225

C.18 *StAdv* attacks against ***CIFAR-10***'s WideResNet-70-16 with *GDMP* purification. Left - original image. Right - *StAdv*. 226

C.19 *StAdv* attacks against ***ImageNet***'s DeiT-S with *DiffPure* purification. Left - original image. Right - *StAdv*. 227

List of Tables

3.1 Top-performing CMs on the ASVspoof2019 LA evaluation dataset with public code and pre-trained models as of October 2021. *In the first row, WAV2VEC is the state-of-the-art CM and its EER is w.r.t. the more comprehensive ASVspoof2021 LA evaluation set. 32

3.2 Attack results against CMs. Bold entries represent the best success results for each system. 35

3.3 Attack results against ASVs. 38

3.4 Attack results (%) against combined ASV-CM-STT deployments. 39

3.5 Attack results against over-telephony-network CMs. 40

4.1 Studied *SOTA* watermarking schemes. 69

4.2 Performance of watermarking schemes against each stage of *UnMarker*. . . 74

4.3 *UnMarker*'s performance against *semantic* schemes with and without optimizable filters. 75

4.4 Comparison with *SOTA* watermark removal attacks. 77

5.1 *AA*-ℓ_∞ performance comparison on ***CIFAR-10*** (ϵ_∞=8/255) under realistic threat models. Metrics include Wor.Rob and MV.Rob under a 10-evaluation protocol. † indicates strategy is *PGD*. 100

5.2 *AA*-ℓ_∞ comparison on ***ImageNet*** ($\epsilon_\infty = 4/255$). 102

5.3 Performance of *LF* attack under *MV*. 105

6.1 Robust accuracy (%) across datasets. For *KEYSHIFT*, we report both **private-mode** and (public-mode) results. 125

6.2 BAN's performance. Acc_{pub} is classification accuracy under public labeling (ground-truth), while Acc_{priv} is the accuracy under π. Successful recovery would decrease Acc_{pub} while increasing Acc_{priv}. 133

6.3 Permutation recovery using universal adversarial perturbations (*UAP*s). A recovery is successful only if the true source class uniquely maps to the target. Across all step budgets the attack never recovers the hidden permutation. . 135

A.1 Architectures, datasets, and EERs of the different ASVs with respect to their evaluation/calibration datasets. 178

A.2 CMs' EERs and their respective thresholds in the phone setting w.r.t. ASVspoof2021/AWSCalibrateCM. EERs below 15% are in bold. 178

A.3 CM performance under additive noise. Bold numbers indicate the highest success rate against each system. 180

A.4 CM performance (%) under transferability. For each attack type, bold numbers correspond to the configuration with the highest success rate against each system. 181

A.5 Independent transformations' success rates against CMs. Bold entries represent the best results for each system. 182

A.6 Results of a simple pre-emphasis attack with a large coefficient (0.97). . . . 182

B.1 Watermark lengths. 187

B.2 Performance of watermarking schemes against basic manipulations. 188

C.1 One-shot AA-ℓ_∞ comparison on ***CIFAR-10*** (ϵ_∞=8/255). † indicates strategy is *PGD*. 205

C.2 AA-ℓ_∞ comparison on ***CIFAR-10*** **under Liu et al. [162]'s protocol (i.e., Fixed *AutoAttack*)** (ϵ_∞=8/255). 207

C.3 AA-ℓ_∞ Performance under various evaluation sample counts 208

D.1 U-Net backbone configurations. 236

D.2 Training loss coefficients. $\lambda_{\text{priv}} = 1.0$ for all datasets. 237

D.3 Private-mode robust accuracy (%) under the *Square* attack with varying query budgets. 240

D.4 Robust accuracy (%) of private-mode *KEYSHIFT* on ***CIFAR-10*** under transferred *AE*s generated against *JBDA* surrogates. 241

Chapter 1

Introduction

Machine learning (*ML*) systems have become deeply embedded in society, shaping how individuals interact with information and services. They offer substantial benefits, advancing education [9], business [50, 43], industrial innovation [211, 107], and scientific discovery [122, 173]. Yet, their capabilities and wide adoption also make them attractive exploitation targets. Incidents of manipulating *ML* models to bypass safeguards, influence outcomes, or cause societal harm are well-documented. For instance, moderation tools deployed by major online platforms have repeatedly struggled against users who slightly modify policy-violating media to evade detection, as observed during the widespread redistribution of altered livestream footage following the Christchurch [201] and Buffalo [202] mass shootings. Meanwhile, advances in generative modeling have enabled the large-scale creation of highly realistic imagery [190], audio [74], and video [107], facilitating political manipulation [5], phishing attacks contributing to substantial financial losses [175], and abusive or non-consensual synthetic media [99]. Significant effort has been devoted to developing countermeasures, including defenses intended to secure *ML* models against unsafe inputs [182, 242], and media-authenticity validation techniques [6, 203, 78, 91]. However, sophisticated strategies that circumvent these protections continue to emerge [243, 130], resulting in an ongoing cat-and-mouse dynamic. These risks are being amplified by the rise of large language models (*LLMs*) and their growing integration into everyday services and decision-making pipelines. Despite safety and alignment efforts, research has demonstrated the ability to circumvent these mechanisms through carefully crafted prompts and interaction strategies [181], raising concerns about the potential of these models not only as targets, but also as facilitators of malicious activities at unprecedented scale. Collectively, these failures suggest misuse is not limited to a particular modality or application, but a broader challenge in securing increasingly capable *ML* systems against skilled and motivated adversaries.

Problem Statement. Reliable deployment of *ML* systems requires defensive mechanisms that preserve utility for benign users while preventing misuse by adversarial actors. In practice, achieving both simultaneously remains challenging. Some certifiable defenses provide formal robustness guarantees, but these guarantees are often valid only under narrow threat models that do not comprehensively capture realistic attacks. Empirical defenses, on the other hand, forgo formal guarantees in favor of robustness against broader classes of real-world threats assessed through experimental evaluation. Yet, they are often subsequently broken by sophisticated attack strategies. This recurring failure pattern suggests the problem is not limited to shortcomings of individual defenses, but reflects a deeper issue in how robustness is conceptualized. Existing systems and defenses fundamentally rely on assumptions about the input distribution, the threat model, or the deployment environment. However, these assumptions are often implicit and insufficiently scrutinized, leading to robustness claims that may not hold under realistic adversarial conditions. This work adopts the view that robustness is inherently conditional: in real-world security settings, robustness guarantees and evaluations become meaningless once the assumptions underlying them no longer hold. Hence, the central question investigated is:

> *How can the evaluation and design of ML security mechanisms account for the assumptions underlying their robustness and their implications in order to provide meaningful real-world security guarantees?*

To operationalize this approach, the thesis introduces *assumption stress-testing*, a framework that combines theoretical analysis with empirical evaluation to systematically i) identify these assumptions, and ii) assess their validity through targeted adversarial strategies. By doing so, it advances the theory and practice of secure *ML* by refining robustness evaluation methodology to expose critical failure modes of widely trusted and state-of-the-art defenses, and informing the design of mechanisms whose security guarantees remain meaningful under realistic adversarial conditions.

1.1 Contributions

Guided by this perspective, a sequence of connected research questions is explored, where insights from one setting inform analysis and design in others. Rather than focusing on a single domain, the work spans multiple paradigms in *ML* security, unified by the goal of stress-testing the assumptions underlying robustness. The primary questions are:

(a) *Is the assumption underlying many ML systems currently deployed in practice—that security can be achieved by simply restricting model access—actually justified?*

(b) *Can we assume strong theoretical arguments necessarily imply real-world robustness?*

(c) *How can the validation of underlying assumptions be systematically formalized?*

(d) *How can practical ML systems be designed to retain meaningful robustness under realistic worst-case conditions and adaptive adversarial pressure?*

To address these questions, the thesis makes the following contributions:

1. *Universal attacks on security-critical ML systems.* This thesis establishes that systems with real-world impact and applications whose security relies on restricted model access can often be bypassed by exploiting properties inherent to the data. This is shown for two distinct paradigms: voice authentication systems and image watermarking schemes designed to support media authenticity. In both settings, the work reveals that these models inevitably rely on identifiable signal-level structures shared across inputs. Furthermore, these structures can be altered through targeted quality-preserving input manipulations, enabling universal black-box attacks without requiring model access or introducing perceptible artifacts.

2. *Systematic evaluation of theoretically motivated defenses.* Principled defenses are increasingly viewed as a promising direction for securing *ML* systems. Thus, this work investigates whether strong formal motivation necessarily translates into real-world resilience, using diffusion-based purification, a state-of-the-art defense against adversarial examples, as a case study to answer in the negative. The investigation reveals a gap between the previously accepted theoretical premise and practical robustness through both formal analysis and empirical assessment under advanced adaptive attacks, while correcting prevalent backpropagation issues in earlier work. Together, these findings substantially challenge prior evaluations.

3. *A unifying framework for evaluating ML security.* This thesis advances *assumption stress-testing* as a general framework for testing and designing secure systems. The framework consists of three primary stages: (i) surfacing and analyzing often implicit assumptions and reasoning about their potential failure modes; (ii) operationalizing these violations as adversarial optimization problems under task-relevant validity constraints; and (iii) characterizing the revealed weaknesses to guide the design of more robust mechanisms. A central challenge addressed throughout the thesis is that assumption violations often do not trivially correspond to well-defined optimization objectives. As such, they require innovative principled formulation through specialized transformations, surrogate objectives, or tailored methodologies to constrain the

structure of adversarial inputs or define the search space. This work instantiates novel applications across diverse security-centric domains, serving as a foundation for broader assumption-aware evaluation in *ML* security.

4. *Assumption-aware design under adaptive threat models.* Building on the identified failure modes, this thesis further showcases how to use *assumption stress-testing* not only as an evaluation philosophy but also as a constructive design framework. It proposes a defense paradigm against adversarial examples that does not merely compete for performance on common benchmarks but explicitly exposes its assumptions and strives to minimize the attack surface arising from their potential violation. By treating adaptive robustness as a central design objective, the resulting mechanism retains meaningful robustness even under active and advanced analysis, marking a promising direction for designing secure systems across diverse *ML* domains.

1.2 Overview

Four projects investigate how assumptions govern the security of *ML* systems across domains. Chapter 2 provides background on machine learning, generative models, content authenticity mechanisms (deepfake detectors and watermarking) and adversarial attacks and defenses.

- **Voice Authentication.** Chapter 3 examines biometric speaker authentication technologies in black-box deployment settings. It develops attacks that circumvent state-of-the-art systems through universal transformations that suppress generative artifacts in synthetic speech, enabling adversaries to impersonate victims while evading deepfake detectors.
- **Image Watermarking.** Chapter 4 investigates the robustness of image watermarking schemes as a countermeasure to harmful AI-generated content. Based on an in-depth analysis, it frames watermark removal as a model-agnostic spectral optimization problem, yielding a generalized attack that compromises recent commercial and academic systems.
- **Adversarial Defense Evaluation.** Chapter 5 revisits diffusion-based adversarial purification defenses under rigorous adaptive evaluation. It identifies evaluation and implementation artifacts that inflate reported robustness and introduces improved attack methodologies that overturn existing security claims.
- **Assumption-Aware Defense Design.** Chapter 6 explores the constructive application of *assumption stress-testing*. It introduces a defense paradigm that explicitly accounts for adaptive adversarial scrutiny at the design level to preserve robustness under advanced attack scenarios.

Chapter 7 summarizes the contributions and outlines directions for future research.

Chapter 2

Background

This chapter provides the background necessary for the analyses developed in this thesis. We review core concepts in machine learning, including classification and generative modeling, with a focus on perceptual domains such as vision and audio. We then introduce mechanisms for content authenticity, namely passive detection and watermarking, followed by an overview of adversarial attacks and defenses. These concepts underpin the studies in subsequent chapters, including attacks on voice authentication systems (Chapter 3), watermarking (Chapter 4), and adversarial defenses (Chapters 5 and 6).

2.1 Machine Learning Foundations

This thesis studies machine learning (*ML*) systems and examines the risks that arise when they are deployed in settings where adversaries may attempt to manipulate or misuse them.

Definition 2.1.1 (Machine Learning)**.** *Machine learning (ML) refers to frameworks that enable systems to infer patterns and make decisions based on data, rather than relying on explicitly programmed rules. By optimizing model parameters with respect to observed data, ML systems can generalize to new inputs and support a wide range of predictive and decision-making tasks.*

2.1.1 *ML* Classifiers

A central class of machine learning models are *classifiers*, which aim to assign inputs to one of a finite set of categories. Given an input $\boldsymbol{x} \in \mathbb{R}^d$, the goal of a classifier is to predict

the corresponding ground-truth label $y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite label space. Such models are widely used in applications including image recognition, speech processing, and content moderation, where decisions must be made based on learned patterns in data.

Formally, a classifier is a function $\mathcal{M} : \mathbb{R}^d \to \Delta^{|\mathcal{Y}|}$ that maps an input to a probability distribution over the label space, where $\Delta^{|\mathcal{Y}|}$ denotes the probability simplex. The predicted label is typically obtained by selecting the class with the highest predicted probability:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \mathcal{M}(\boldsymbol{x})_y.$$

Training Objective. Classifiers are commonly trained using supervised learning, where the model is optimized over a dataset of input-label pairs. Let θ denote the model parameters. The training objective is to minimize a loss function that measures the discrepancy between the predicted and true labels:

$$\min_{\theta} \ \mathbb{E}_{(\boldsymbol{x},y)} \left[\ell\big(\mathcal{M}_\theta(\boldsymbol{x}), y\big) \right].$$

A widely used choice is the cross-entropy loss, defined as:

$$\ell(\mathcal{M}(\boldsymbol{x}), y) = -\sum_{k=1}^{|\mathcal{Y}|} y_k \log\big(\mathcal{M}(\boldsymbol{x})_k\big),$$

where y_k denotes the one-hot encoding of the true label.

The ability of a classifier to generalize beyond the training data—i.e., to correctly predict labels for previously unseen inputs—is central to its effectiveness in real-world applications.

2.1.2 Generative & Diffusion Models

In addition to predictive models such as classifiers, machine learning systems are increasingly used to generate data that resembles a target distribution. These *generative models* aim to learn this underlying data distribution and produce new samples that are consistent with the training data.

Formally, given a distribution $p(x)$ over $\mathbb{R}^d$, a generative model seeks to approximate this distribution and enable sampling from it. Unlike classifiers, which learn mappings from inputs to labels, generative models aim to capture the structure of the data itself. Recent advances in generative modeling have led to highly expressive models capable of producing realistic images, audio, and text.

We focus on generative models for perceptual data, primarily in the visual and audio domains, which form the basis for the deepfake threats, watermarking, and robustness analyses in Chapters 3, 4 & 5. While generative modeling also plays a central role in text generation, such models are beyond the scope of this thesis.

Overview

Generative models encompass several paradigms, including generative adversarial networks (*GAN*s) [86], variational autoencoders (*VAE*s) [133], and diffusion models [227, 230]. While these approaches differ in their training objectives and architectures, they share the common goal of approximating complex data distributions. *GAN*s, for instance, learn to generate data through an adversarial game between a generator and a discriminator, while *VAE*s model data via latent variable inference. Diffusion models, in particular, have achieved state-of-the-art performance in high-dimensional data generation [63, 246].

Although all of these generative paradigms appear in the systems considered throughout the thesis, Chapter 5 examines diffusion-based adversarial defenses [182, 250] in detail, which necessitates a deeper understanding of their underlying mechanisms. We therefore provide a more detailed description of diffusion models in the following subsection.

Diffusion Models

Diffusion models (*DM*s) [227, 230] learn to model a distribution p on $\mathbb{R}^d$ by reversing the process that diffuses inputs into noise. *DM*s involve two stochastic processes. The forward pass converts samples into pure Gaussians, and is governed by the following *SDE* for an infinitesimal step $dt > 0$:

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)dt + g(t)d\boldsymbol{w}. \tag{2.1}$$

eq. (2.1) describes a stochastic integral whose solution up to $t^* \in [0, 1]$ gives $\boldsymbol{x}(t^*)$. Here, $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R} \longrightarrow \mathbb{R}^d$ is the drift, $\boldsymbol{w} : \mathbb{R} \longrightarrow \mathbb{R}^d$ is a Wiener process, and $g : \mathbb{R} \longrightarrow \mathbb{R}$ is the diffusion coefficient. We focus on **VP-SDE**. Yet, our insights generalize to all *DM*s (see [230] for a review). In **VP-SDE**, $\boldsymbol{f}(\boldsymbol{x}, t) = -\frac{1}{2}\beta(t)\boldsymbol{x}(t)$ and $g(t) = \sqrt{\beta(t)}$, where $\beta(t)$ is a noise scheduler outputting small positive constants. These choices yield the solution:

$$\boldsymbol{x}(t^*) = \sqrt{\alpha(t^*)}\boldsymbol{x} + \sqrt{1 - \alpha(t^*)}\boldsymbol{\epsilon} \tag{2.2}$$

for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ and $\alpha(t) = e^{-\int_0^t \beta(s)\, ds}$. With proper parameters, we have $\boldsymbol{x}(1) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$. Thus, a process that inverts *eq. (2.1)* from $t^* = 1$ to 0 allows generating samples in p from

random noise. Due to Anderson [17], the reverse pass is known to be a stochastic process with:

$$d\hat{\boldsymbol{x}} = [\boldsymbol{f}(\hat{\boldsymbol{x}}(t),\, t) - g^2(t)\nabla_{\hat{\boldsymbol{x}}(t)} \log p_t(\hat{\boldsymbol{x}}(t))]dt + g(t)d\bar{\boldsymbol{w}}. \tag{2.3}$$

Defining $\hat{\boldsymbol{x}}(t^*) := \boldsymbol{x}(t^*)$, the process evolves from t^* to 0 with a **negative** time step dt and reverse-time Wiener process $\bar{\boldsymbol{w}}(t)$. Let $p(\boldsymbol{x})$ be the probability of $\boldsymbol{x}$ under p, and $p_{0t}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ the conditional density at t given $\boldsymbol{x}(0) = \boldsymbol{x}$. Then, the marginal density is given by $p_t(\tilde{\boldsymbol{x}}) = \int p(\boldsymbol{x})p_{0t}(\tilde{\boldsymbol{x}}|\boldsymbol{x})d\boldsymbol{x}$, where $p_0 \equiv p$. Solving *eq. (2.3)* requires the **score** $\nabla_{\hat{\boldsymbol{x}}(t)} \log p_t(\hat{\boldsymbol{x}}(t))$, which can be approximated via a trained model $\boldsymbol{s}_\theta$ s.t. $\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}(t), t) \approx \nabla_{\hat{\boldsymbol{x}}(t)} \log p_t(\hat{\boldsymbol{x}}(t))$ [230], yielding:

$$d\hat{\boldsymbol{x}} = -\frac{1}{2}\beta(t)[\hat{\boldsymbol{x}}(t) + 2\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}(t), t)]dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}}. \tag{2.4}$$

As no closed-form solution exists, the process in practical applications runs iteratively over **small but discrete** negative steps dt (typically, $|dt| = 0.001$). Starting from any t^*, $d\hat{\boldsymbol{x}}$ is calculated at each $i = |\frac{t}{dt}|$, until $t = 0$. This *continuous-time DM* describes a stochastic integral (despite the discretized implementations). An alternative, *Denoising diffusion probabilistic modeling* (**DDPM**) [106, 227], considers a *discrete-time DM*. The two are equivalent (see Appendix C.1).

2.2 Content Authenticity Mechanisms

The increasing capability of generative models has enabled the creation of highly realistic synthetic content. While these models offer significant benefits, they also enable the generation of *harmful deepfakes*—synthetic media that convincingly imitates real data, including images, audio, and video—used to mislead, impersonate individuals, or spread damaging and non-consensual content.

Considerable effort has been devoted to developing mechanisms for assessing the authenticity and provenance of such content across multiple modalities. This thesis focuses on the visual and audio domains. Two primary approaches have been studied: *passive detection*, which identifies synthetic content based on its characteristics, and *watermarking*, which embeds identifiable signals into generated data. Passive detection forms the basis of the analysis in Chapter 3, particularly in the context of audio, while watermarking is studied in the visual domain in Chapter 4. The limitations of passive detection also serve as a key motivation for watermarking, and are hence discussed in both contexts (vision and audio).

2.2.1 Passive Detection

Passive detection methods [118, 150, 81, 297, 6, 235, 172] were among the earliest approaches proposed to address the threat of deepfakes. These methods are studied across modalities—including images, audio, and video—and typically involve training classifiers to distinguish between real and synthetic data based on measurable distributional differences.

Such systems rely on detecting semantic inconsistencies [109, 169] or low-level artifacts [118, 81, 205, 168, 6, 235] introduced during the generative process. For example, in the visual domain, detectors may exploit discrepancies in textures, lighting, or structural coherence that are difficult for generative models to reproduce faithfully. In audio (and video), these systems, commonly known as spoofing countermeasures (CMs), may additionally leverage *liveness* cues in time-varying signals to distinguish genuine recordings from replayed or synthesized speech [172].

Given that such detectors are the primary focus of Chapter 3, where we study attacks on *ML*-based voice authentication systems, a more detailed discussion of audio deepfake detection methods is deferred to that chapter.

2.2.2 Watermarking

Our focus is on watermarking for visual content, which is studied in detail in Chapter 4. While passive detectors remain widely used in practice, their effectiveness depends on the presence of identifiable artifacts or inconsistencies. As generative models improve, they are increasingly capable of producing high-quality outputs that reduce or eliminate such signals, making detection more challenging. Furthermore, passive detectors may be vulnerable to adversarial manipulations that can obscure or remove detectable features [69, 27, 129] as we also show in Chapter 3, and often struggle to generalize to content generated by previously unseen models [49].

These limitations have motivated more proactive approaches. A key observation underlying this paradigm is that, in practice, generating high-quality synthetic content requires substantial data, computational resources, and engineering effort. As a result, most adversaries wishing to misuse deepfakes to cause harm are unable to train state-of-the-art generative models independently and instead rely on publicly accessible systems provided by major *ML* providers. This reliance creates an opportunity to enforce control at the source of content generation. Rather than attempting to identify fake content based on potentially unreliable artifacts, providers intentionally embed identifying signals into the outputs of

their models before releasing them. If a suspicious image later surfaces, designated detectors can be used to verify whether it was generated by a particular system.

A key requirement for effective watermarking is that the generated outputs remain semantically or perceptually consistent with the intended content, while encoding a signal that can be reliably detected. At a high level, an (image) watermarking scheme consists of two components:

- **Encoder** $\boldsymbol{x}_w \leftarrow \mathcal{E}_\theta(\boldsymbol{c}, \boldsymbol{w})$: Maps a conditioning input $\boldsymbol{c}$ (e.g., an image, a text prompt, or latent variables) and an *optional* watermark $\boldsymbol{w}$ to an output sample $\boldsymbol{x}_w$.
- **Detector** $s \leftarrow \mathcal{D}_\phi(\boldsymbol{x}, \boldsymbol{w})$: Maps a sample $\boldsymbol{x}$ and an *optional* watermark $\boldsymbol{w}$ to a detection score $s \in \mathbb{R}$, corresponding to a degree of confidence with respect to the presence of the watermark $\boldsymbol{w}$ in $\boldsymbol{x}$. In practice, this score is compared against a threshold to produce a binary decision (watermarked or not), where the threshold is typically chosen to lower the false positive rate. The specific decision procedures depend on the watermarking scheme and are discussed in §4.6.1.

Notes: In this formulation, the watermark input $\boldsymbol{w}$ is optional. In some schemes [282, 281], the encoder and detector are trained to embed and recognize a fixed watermark determined at training time, in which case $\boldsymbol{w}$ is implicit and need not be provided explicitly. Additionally, when watermarking is integrated into the generation process (see below), the generator itself could serve as the encoder and $\boldsymbol{w}$ may be omitted, resulting in the generation of non-watermarked content. We denote the encoder and detector as $\mathcal{E}_\theta$ and $\mathcal{D}_\phi$ to allow for parameterized (e.g., learned) implementations, although these components may also be fixed or heuristic.

The above abstraction captures a wide range of watermarking schemes:

General-Purpose (Post-Hoc) Schemes. Here, the conditioning input corresponds to an image ($\boldsymbol{c} = \boldsymbol{x}$), and the encoder produces a visually similar watermarked version $\boldsymbol{x}_w$. Early watermarking methods fell under this category but were primarily heuristic, manipulating the input images' spectra to form a signature in select frequencies [24, 200, 224]. Yet, they have been shown to be vulnerable to removal via simple post-processing transformations [120, 78]. More recent *general-purpose* methods address these limitations by learning dedicated encoding and detection mechanisms [303, 236], resulting in improved robustness. Thus, the post-hoc schemes this thesis considers belong to such learned, more robust variants.

Generator-Specific (Generative) Schemes. In these approaches [282, 281, 164, 78, 260], the encoder is integrated into the generation pipeline, and $\boldsymbol{c}$ may represent a prompt or latent

variables (potentially with a random seed that controls the stochastic generation). The encoder operates as part of—or on top of—the generative model, influencing the synthesis to embed the watermark. Depending on the design, such schemes [260, 78] may allow for both watermarked and non-watermarked generation, for example by conditioning on the presence or absence of a watermark input. In such settings, using the same conditioning—including the underlying randomness—ensures watermarked ($\boldsymbol{x}_w$) and non-watermarked ($\boldsymbol{x}$) outputs correspond to the same content and generation trajectory, enabling meaningful notions of similarity.

2.3 Adversarial Attacks and Defenses

As machine learning systems are increasingly deployed in real-world settings, they become attractive targets for malicious actors seeking to exploit them for financial, personal, or societal gain. In the machine learning literature, attempts to manipulate model behavior in such settings are commonly referred to as *adversarial attacks*. These attacks aim to construct inputs that induce incorrect or unintended behavior from a model while remaining consistent with a task-relevant notion of validity.

In this section, we provide the necessary background on adversarial attacks against vision and audio models (classifiers), which are studied throughout the thesis across a range of machine learning models and tasks, including attacks on biometric authentication systems and deepfake detection mechanisms (Chapter 3), watermarking schemes (Chapter 4), and image classifiers (Chapters 5 & 6).

2.3.1 Adversarial Examples

In perceptual domains (e.g., vision and audio), the malicious inputs generated by such attacks are commonly referred to as *adversarial examples* [233, 87]—samples that remain perceptually or semantically consistent with legitimate data while causing the model to produce an incorrect output. Traditionally, adversarial examples refer to inputs that exploit weaknesses in a model's learned decision function under constraints that ensure they remain indistinguishable from the benign samples from which they were crafted. Formally:

Definition 2.3.1 (Adversarial Examples)**.** *Let $\boldsymbol{x} \in \mathbb{R}^d$ denote an input with true label y, and let $\mathcal{M}$ be a classifier. Adversarial examples (AEs) are defined as solutions to the following constrained optimization problem:*

$$\boldsymbol{x}_{adv} = \underset{\mathcal{D}(\boldsymbol{x}', \boldsymbol{x}) \leqslant \epsilon_{\mathcal{D}}}{\arg\min} \; \ell(\mathcal{M}(\boldsymbol{x}'),\ y)$$

where ℓ is a loss function that encourages misclassification, $\boldsymbol{\mathcal{D}}$ is a distance metric that enforces similarity between the original and perturbed inputs, and $\epsilon_{\boldsymbol{\mathcal{D}}}$ bounds the allowed perturbation magnitude.

Targeted vs. Untargeted Attacks

The formulation in Definition 2.3.1 corresponds to *untargeted* attacks, where the goal is to induce any incorrect prediction. In contrast, *targeted* attacks aim to force the model to predict a specific target label $y_t \neq y$. This is achieved by modifying the objective to encourage misclassification toward y_t, typically by replacing y with y_t in the optimization problem and updating the loss function accordingly (e.g., flipping its sign). Targeted attacks are often more restrictive but can be significantly more damaging in practice, as they enable controlled manipulation of model outputs.

Similarity Metrics

The constraint $\boldsymbol{\mathcal{D}}(\boldsymbol{x}', \boldsymbol{x}) \leqslant \epsilon_{\boldsymbol{\mathcal{D}}}$ enforces similarity between the original input and its adversarial counterpart, and plays a central role in defining adversarial examples. While an attacker could trivially induce misclassification by providing arbitrary inputs, such inputs would typically fall outside the distribution of valid data and may be rejected by downstream systems or human inspection. In many practical settings, particularly in security-critical applications, inputs must satisfy semantic or perceptual constraints—for example, they may be captured from live sources, verified for plausibility, or subject to human scrutiny. As a result, adversarial attacks are typically constrained to remain close to a reference input as a methodical way of ensuring that the perturbed sample remains valid within the domain.

The most common choices for $\boldsymbol{\mathcal{D}}$ are ℓ_p norms, which have been extensively studied in the adversarial robustness literature [53, 141, 32]. These norms provide convenient mathematical formulations for bounding perturbation magnitude. However, ℓ_p constraints do not fully capture perceptual similarity, particularly in high-dimensional domains such as images and audio. As a result, alternative metrics have been proposed, including perceptual similarity measures that better align with human judgment [143, 271]. The choice of similarity metric plays a critical role in defining the space of valid adversarial examples and can significantly influence both attack effectiveness and evaluation outcomes.

Threat Models

Adversarial attacks are commonly analyzed under different threat models, which specify the attacker's knowledge and capabilities:

- **White-box** settings: The attacker has full access to the model, including its parameters and gradients, enabling direct optimization of the objective.
- **Black-box** settings: The attacker has limited access, typically only to the output labels or probability vectors, and must estimate useful optimization trajectories through repeated queries or surrogate models to which white-box access is available to craft transferable perturbations.
- **No-box** settings: The most restrictive setting, sometimes referred to as the *query-free* setting, where the attacker is assumed to not have the ability to interact with the target model, requiring attacks to rely on surrogate models or exploit general properties of the system or data distribution.

These access-level threat models play a central role in evaluating robustness: attacks that succeed under stronger or more restrictive assumptions are generally considered more realistic and indicative of practical vulnerabilities.

In practice, however, adversaries rarely interact with a model in isolation. *ML* systems deployed in high-stakes settings—where incorrect or manipulated outputs can have severe consequences—are typically accessed through operational interfaces that impose additional constraints on interaction. That is, even when access is limited to model outputs only, as in black-box settings, deployed services may restrict the nature of that feedback. We refer to settings governed by such operational constraints as **security-critical deployments**:

Definition 2.3.2 (Security-Critical Deployments)**.** *A* ***security-critical deployment*** *is an ML system operating in a setting where the consequences of failure are severe (e.g., biometric authentication, fraud detection, access control, or autonomous driving). Beyond access-level threat models, such deployments* ***may be*** *subject to operational safeguards that further constrain how adversaries may interact with the deployed service.*

These safeguards serve as an additional layer of security to complicate, deter, or hinder successful attack optimization and often involve bounding the number or rate of allowed queries to the model, step-up verification under suspicious activity, requiring user input within strict time limits (e.g., in authentication tasks), or information disclosure controls such as returning hard labels only rather than confidence scores.

These constraints need not be separate from the access assumptions above; in many settings they interact with them directly. For instance, severely limited query budgets together with hard-label-only feedback can, in some cases, render attacks that depend on extensive model interaction impractical, effectively pushing adversaries toward query-free (no-box) regimes despite nominal black-box access, as demonstrated in Chapter 3.

Adversarial Example Generation (Adversarial Attacks)

The process of generating an adversarial example is commonly referred to as an adversarial attack. The constrained problem in Definition 2.3.1 is typically solved using iterative optimization methods. In white-box settings, gradient-based approaches are widely used, where perturbations are constructed by following directions that optimize the misclassification loss. Formally, we define this standard gradient-based approach as follows:

Definition 2.3.3 (Standard Gradient-Based Attack)**.** *Given classifier $\mathcal{M}$ and loss $\ell(\mathcal{M}(\boldsymbol{x}), y)$ from Definition 2.3.1, the standard gradient-based attack iteratively updates the adversarial example by computing gradients of the loss with respect to the input:*

$$\nabla_{\boldsymbol{x}} \ell(\mathcal{M}(\boldsymbol{x}), y),$$

and modifying $\boldsymbol{x}$ in directions that optimize ℓ while enforcing the similarity constraint imposed by $\boldsymbol{\mathcal{D}}$.

This formulation encompasses a broad class of attack instantiations that vary in their optimization procedures, perturbation parameterizations, step sizes, or projection mechanisms (e.g., *PGD* [141], *C & W* [32], *AutoAttack* [53]), while still remaining within this standard gradient-based form. In black-box settings, gradient-free methods or substitute models are often employed to approximate the required gradients.

While adversarial attacks are traditionally crafted in this manner, this approach is not exhaustive. In practice, *AE*s may also be constructed through surrogate objectives [130], structured transformations, or domain-specific manipulations that preserve perceptual or semantic validity without directly optimizing a loss function [1].

Recent advances in generative modeling have also enabled attacks that leverage learned data distributions to construct adversarial inputs [298, 214, 15], for example by guiding generative processes to produce semantically valid samples that induce misclassification [41]. These approaches, commonly referred to as *unrestricted adversarial examples (UAEs)* or *regeneration attacks*, are often studied separately from traditional adversarial examples, as they do not enforce explicit similarity requirements with respect to a fixed reference input.

2.3.2 Adversarial Defenses

Adversarial defenses aim to improve the robustness of machine learning systems against adversarial manipulation. These methods seek to ensure that model predictions remain stable under input perturbations that preserve the semantic or perceptual validity of the data. These defenses fall under the following classes:

Certified Defenses

Certified defenses provide formal robustness guarantees, typically via randomized smoothing or bound propagation [92, 47, 217, 176, 147]. While offering valuable analytical insight into model behavior, such bounds typically incur significant overhead and apply to restricted threat models (e.g., small ℓ_2 perturbations). Thus, extensive work studies **empirical defenses** aiming for efficiency and robustness to realistic attacks. We categorize those broadly as follows:

Adversarial Training (AT)

Adversarial training augments training data with AEs, encouraging the classifier to correctly label them [167]. It is regarded among the most successful methods to enhance robustness, with numerous works proposing improved networks, training and data augmentation schemes [207, 257, 287, 56, 22, 198]. Yet, AT primarily provides robustness to threats encountered during training—most commonly ℓ_p-bounded—and is known to be vulnerable to unseen attacks [143, 271].

Preprocessing Defenses

These methods apply input pre-classification operations to detect, suppress, or disrupt attacks, employing randomized transformations [96, 273] and compression [58, 275], while later methods utilize manifold projections [105] and spectral decompositions [66].

Despite their initially perceived robustness, many such methods often derive their effectiveness from *gradient masking*, where the defense obscures or distorts informative gradients without eliminating the underlying vulnerability. This can lead to a false sense of security when evaluated with standard attack methods.

A key challenge in evaluating these defenses arises from *adaptive attacks*, which explicitly account for the defense during attack construction by analyzing its mechanism and tailoring

attacks to circumvent it. In such settings, attackers may differentiate through the entire defended system or optimize an expected objective when the defense is stochastic, for example using Expectation Over Transformation (EOT) [20]. Some defenses attempt to obfuscate gradients by introducing non-differentiable components; however, these can often be bypassed using backward pass differentiable approximations (BPDA) [20]. In other cases, specialized loss functions or similar techniques are employed to explicitly account for specific components of the defense during optimization. As a result, the above defenses and many similar approaches [194, 270, 278, 21, 171, 108, 213] were later broken by adaptive attacks [243, 20, 31].

A subclass of preprocessing, namely *purification*, uses generative models to project adversarial inputs onto the natural data manifold [218, 103, 229], with *diffusion-based purification (DBP)* [182, 250] recently gaining significant attention [272, 35, 288, 38, 267, 39, 286, 192]. Yet, later analyses, including those presented in Chapter 5, suggest that the perceived robustness of such methods also stems from limitations in adaptive evaluation [131].

Chapter 3

Breaking Security-Critical Voice Authentication

This chapter is adapted from work published at the IEEE Symposium on Security and Privacy (S&P) 2023 [129]. It studies voice authentication systems used in high-stakes settings (e.g., banking) through the lens of *assumption stress-testing*, targeting a widely upheld deployment assumption: that hiding model internals and restricting interaction naturally constrains attack feasibility. We show this assumption fails by asking whether model interaction is necessary at all, and answering in the negative by demonstrating that spoofed (fake) speech can be systematically manipulated to bypass authentication systems despite their use of dedicated deepfake detectors (spoofing countermeasures). Rather than attacking these detectors directly, we identify the potential artifacts such systems rely on to distinguish human from synthetic speech and develop model-agnostic transformations that remove them, causing detectors to fail. Applying these transformations thus enables universal bypass of the full authentication pipeline with up to 99% success within only a few attempts (up to six), without requiring access to the target models. Beyond academia, this work led to security patches in deployed systems, collaborations with a leading voice-authentication provider, and a patent filing.

3.1 Motivation

Automatic speaker verification systems (ASVs) are widely used for authentication where a claimed identity is verified by comparing features extracted from a given audio sample against

a "voiceprint" obtained from previously collected recordings. ASVs have gained popularity, primarily due to their convenience and increased security compared to passwords, becoming the core component of voice authentication (VA)—a constantly growing multi-billion dollar market [238].

VA has been deployed in high-stakes settings such as banking (e.g., Citibank [94] and First Direct [79]). Nuance [185] (recently acquired by Microsoft for $19.7B), Aculab [4], and similar products are widely used for authentication in call centers or for enabling transactions using smartphone apps [186, 187]. In light of the mass adoption of VA and boastful statements ("No one else has a voice just like you" [237]) of its vendors, it becomes imperative to evaluate its security under realistic threat scenarios.

Several attacks against ASVs have emerged [51]. Yet, the popularity of VA as a robust authentication platform for sensitive tasks is still on the rise. The reason is that no existing attack has demonstrated a proven ability to circumvent VA under the strict deployment constraints enforced in these environments—see §3.2.4. Spoofing attacks (or deepfakes), such as speech synthesis (SS) [223] or voice conversion (VC) [239, 177], have shown great potential in fooling ASVs via fake audio generated in the victim's voice [261]. However, this threat has been known for years [277], leading to the development of spoofing countermeasures (CMs)—complementary systems deployed side-by-side with ASVs, whose task is to detect spoofed (fake) speech. CMs are now being deployed by VA industry leaders [191], such as Nuance [183] and ID R&D Live [114]. The high accuracy achieved by these systems [115] can hinder spoofing attacks. On the other hand, known, non-spoofing, adversarial attacks against ASVs (and automatic speech recognition systems—ASRs) are not well-suited for security-critical deployment constraints as they require whitebox access [157, 33, 138], are query-inefficient (hundreds or thousands of interactions with the target model) or time-inefficient [10, 42, 300, 37, 166, 72], or are untargeted (cannot impersonate a *specific* user) [1]. Finally, while there have been several attempts to adversarially attack VA platforms by generating spoofed speech first and then adversarially modifying it to fool CMs [160, 295, 293, 292, 85], which is similar to the approach we take, these share the same limitations with the adversarial attacks above.

This chapter presents the first practical attack on VA under security-critical deployment constraints. This is the only attack so far that meets all limitations of such environments—fully blackbox, model agnostic, real-time, query-efficient and, most importantly, targeted (the attacker can impersonate any chosen victim). Furthermore, our attack accounts for all entities present in this strictest form of real-world VA, which typically also utilizes text-prompted authentication: instead of fixed phrases attackers can record the user uttering and replay later, the user is prompted with a random phrase to repeat upon each access attempt. The entities the attacker must circumvent *simultaneously* therefore become the ASV,

CM, and a speech-to-text (STT) unit verifying the random phrase was repeated correctly. We formalize the operational constraints characterizing security-critical deployments in this domain in §3.2.4. Our attack exploits the known vulnerability of ASVs to spoofing attacks [261] and *mainly targets the CMs*. Unlike the few existing adversarial attacks on CMs listed above, our attack employs signal-processing (optimization-free), model-agnostic transformations to grant spoofed speech the ability to universally bypass these systems, while preserving the readily-existing voiceprint and textual content to still bypass ASVs and STTs. As a consequence, users will lose trust in VA, and vendors will have to invest resources in improving its security.

Overall, this work makes the following contributions:

- **Our attack can circumvent several state-of-the-art VA platforms implementing the authentication protocol in its strictest form and is fully black-box, query-efficient, real-time, model-agnostic, and targeted.** To commit fraud or identity theft, attackers aiming to exploit such security-critical deployments of VA (see §3.2.4) must mount *targeted* attacks and will not have access to the model's internals nor will they be able to query it repeatedly. A key contribution of this work is the ability to manipulate spoofed audio generated in the voice of any victim user through signal-level transformations that can universally make any CM accept it, without degrading the quality of the sample. This optimization-free approach circumvents the lack of adversarial transferability in the audio domain and the query and time limitations hindering traditional attacks, enabling, for the first time, the execution of such targeted (user-specific) attacks in a model-agnostic manner.

- **Our attack is effective against the full VA stack.** Although our attack targets CMs, the ultimate goal is to bypass the entire VA stack. Hence, transformed samples must bypass the ASV, which follows from the well-documented vulnerability of ASVs to spoofed speech, given that our transformations are subtle enough to preserve this property. Similarly, the contents of the transformed samples must match the text expected by the STT. Therefore, we design our attacks such that the changes to the spoofed speech are minimal. We verify these assumptions through large-scale experiments involving 14 state-of-the-art CMs, five renowned ASVs (including a commercial system—Amazon Connect Voice ID [13]), and two commercial STTs [89, 174], proving them all vulnerable. With only up to six authentication attempts, the attacker achieves a success rate against the full authentication stack as high as 99%. Our user study further proves that even a human inspecting the outputs of our algorithm will be fooled. Finally, we establish that our attack remains effective even when executed over the phone network. To the best of our knowledge, this is the first-ever *targeted* adversarial attack over the phone.

- **Robustness to various defenses.** We analyze the potential of different defenses in mitigating our attack. While we find common adversarial defenses limited, other adaptive approaches seem more promising. We evaluate these techniques and demonstrate that, when combined, they can degrade the attack's performance. However, the attack remains effective even after this degradation occurs.

- We make our source code publicly available to aid other researchers in the construction of future defenses[1].

3.2 Background

3.2.1 Automatic Speaker Verification (ASV)

At the core of ASVs lies the voiceprint—a set of unique features of an individual found in their voice. This voiceprint captures physical factors like the shape and size of the vocal tract and larynx, and a "behavioral signature" consisting of one's accent, rhythm, pronunciation and more [301]. This led to investigating voiceprints as biometrics.

ASVs operate in two phases: enrollment and verification. During enrollment, the user supplies speech samples used to extract their voiceprint, which serves as their identifying signature onward. Upon future verification attempts, the user's identity is verified via a provided speech sample $\boldsymbol{x}$ together with a claimed user identity U_{ID} to whom the sample supposedly belongs. The sample is checked against the voiceprint outputting a decision (accept/reject). The notation $DB(UID)$ refers to a database query on the server side with the provided identity that retrieves the voiceprint, and $ASV(\boldsymbol{x},\ DB(U_{ID}))$ indicates that the ASV accepts $\boldsymbol{x}$ with respect to the voiceprint of U_{ID}. Verification can follow one of three schemes: 1) Text-dependent—a fixed phrase is repeatedly used, 2) Text-independent—the user may provide any random phrase, or 3) Text-prompted—the system requires a specific random text from the user to be spoken. The second and third options are more secure due to their robustness to replay attacks.

Researchers have studied various sets of input features for ASVs, including the Mel-frequency Cepstral Coefficients (MFCC) and Log Power Magnitude Spectrum (LPMS). ASV architectures vary significantly, but mainly fall under three categories: GMM i-vector systems, x-vectors, and DNN end-to-end approaches that operate on raw waveforms [152].

[1] https://github.com/andrekassis/Breaking-Security-Critical-Voice-Authentication

3.2.2 Spoofing Countermeasures (CMs)

Several spoofing attacks could compromise ASVs [261], and those can be classified into four categories: 1) Mimicking [145]—manipulating one's voice (without machines) to sound like the victim, 2) Speech synthesis (SS) [223]—using technology to produce samples in the victim's voice, 3) Voice conversion (VC) [134]—using technology to convert one's voice into the victim's, and 4) Replay [268]—replaying a previously-recorded sample of the victim. ASVs are robust to mimicking attacks [59] and replay can be prevented using text-independent/text-prompted schemes. However, VC/SS attacks pose serious threats [261].

The need for reliable authentication therefore led to the emergence of spoofing countermeasures (CMs) [6, 277]. These are classifiers solving the problem of distinguishing between human (bonafide) and machine (spoofed) speech. The literature distinguishes between active and passive CMs [172]. Active methods (e.g., VoiceLive [289], VoiceGesture [290], EchoVib [16]) rely on heuristics and sensory data to capture liveness cues that accompany human speech, such as articulatory gestures or the shape of the vocal tract. Passive methods focus on measurable differences in the audio waveform itself that differentiate human from machine speech [6, 172, 235]. Active methods are much less practical since they are sensitive to sensor and microphone placement and are device-dependent, while passive methods can be integrated into existing VA software [172], which is done by pioneers in the field such as Nuance [183] and ID R&D Live [114]. Our focus is, thus, on passive CMs.

Terminology note. The distinction between *active* and *passive* CMs used here follows the speech processing literature and refers to whether liveness cues are actively elicited or inferred from the signal. Both categories fall under the broader class of *passive deepfake detection methods* defined in §2.2.1, in that they operate on measurable features of the input rather than relying on proactive mechanisms such as watermarking.

Not every VA system represents a security-critical deployment. For instance, Siri and Google Assistant neither enforce strong security measures nor use CMs. However, leading VA providers, such as Nuance, Aculab, ID R&D Live, and others are explicit about deploying CMs to defend against spoofing as integral components of their systems [114, 183, 4] in order to secure sensitive resources (e.g., bank accounts). Thus, successful attacks in such high-stakes settings must circumvent CMs.

3.2.3 Speech-to-text Systems (STTs)

Known also as automatic speech recognition systems (ASRs), STTs are used to transcribe spoken phrases into text. In the context of VA, STTs are coupled with ASVs to implement text-independent/text-prompted schemes and defend against replay attacks. The STT is used to transcribe the user's input (sample) and the transcription output is examined against the expected text. The STT accepts the authentication attempt only if the two match. Given a spoken phrase $\boldsymbol{x}$ and text t, the notation $STT(\boldsymbol{x},\ t)$ indicates that the STT accepts $\boldsymbol{x}$ w.r.t. t.

3.2.4 VA in Security-Critical Deployments

This subsection instantiates the notion of security-critical deployments introduced in **Definition** 2.3.2 for voice authentication. In high-stakes domains such as banking, voice authentication is deployed under strong operational constraints as it manages access to highly sensitive resources. These constraints often exist by default and are as follows:

Limited number of failed attempts. Multiple failed attempts lead the target to stop accepting further attempts. Attacks on (voice) authentication platforms that ignore this requirement are invalid. This is standard practice in security-sensitive applications. For instance, credit cards are invalidated if the user enters their PIN incorrectly three times, and Apple locks iPhones after six incorrect passcodes.

Limited response time. This is yet another typical security measure. The user (attacker) has to generate authentication responses and transmit them within seconds.

No access to system internals. The system is blackbox returning only decisions (accept/reject), as assumed in research [300] and done in practice [114, 183]. We do not assume more permissive settings where systems reveal output probabilities.

Success under limited time and attempt constraints sets us apart from all previous blackbox attacks on VA (see §3.10), while those attacks that cannot operate in blackbox settings are not practical. Our attack is fully blackbox, takes 4s on average, and is highly successful with only three to six attempts.

Text-prompted/independent scheme. The safer form of VA uses text-prompted/independent schemes to mitigate replay attacks. Some vendors may use text-dependent schemes with fixed phrases. However, even here, there are CMs that detect replayed speech [6]. Hence, we assume that simply replaying a recording is not a valid attack and assume the more restrictive text-prompted/independent setup.

All in all, to successfully pass an authentication attempt under a security-critical VA deployment with a phrase $\boldsymbol{x}$ given identity U_{ID} and text t, the following must hold:

$$ASV(\boldsymbol{x},\ DB(U_{ID})) \wedge CM(\boldsymbol{x}) \wedge STT(\boldsymbol{x},\ t).$$

3.3 Threat Model

3.3.1 Definitions

Attacker. The malicious actor aiming to bypass a security-critical VA platform via adversarial or spoofed audio to pose as the victim.

Victim. The user protected by the VA system whose identity is at risk of being stolen by the attacker.

Target. The VA system (or any of its components—ASV, CM, or STT) the attacker wishes to fool.

3.3.2 Attack Channels

Here we list the common channels that users (attackers) typically use to communicate with a remote security-critical VA deployment.

Designated-app. The user (attacker) issues a transaction using an app (e.g., a banking app) and authenticates it using their voice (instead of a password) [183]. The authentication phrase is passed to the server over TCP (i.e., reliably, without packet loss, jitter or lossy encodings [185]).

Over-telephony. The user (attacker) calls the service provider and interacts with the interactive voice response (IVR) system, which uses VA for identity verification [183]. The input is transmitted over the phone, introducing additional challenges, as attacks have to withstand the noisy medium (see §3.7).

3.3.3 Attacker's Goals, Knowledge, and Capabilities

Goals. The attacker's goal is to produce an audio sample capable of bypassing a security-critical VA deployment. Based on the components we introduced in §3.2.4, the attacker is

concerned with realizing an algorithm $\mathcal{A}$, whose inputs are text prompt t and the victim identity U_{ID}, and whose output is an adversarial $\boldsymbol{x}_{adv} = \mathcal{A}(t,\ U_{ID})$, s.t:

$$ASV(\boldsymbol{x}_{adv},\ DB(U_{ID})) \wedge CM(\boldsymbol{x}_{adv}) \wedge STT(\boldsymbol{x}_{adv},\ t)$$

The system accepts verification phrases in one of the two methods described in 3.3.2. The attack must comply with the protocol described in §3.2.4.

Optional goal: We optionally require that the samples provided be robust to human inspection. In security-critical deployments, authentication attempts can be recorded for future audit purposes (e.g., disputing suspicious transactions) [185], during which the recordings may be inspected by humans. Passing the human judge's (HJ) inspection, whose task is to listen to the sample **in case of an investigation** and verify whether it in fact comes from the victim, is indicated as $HJ(\boldsymbol{x}_{adv},\ U_{ID})$.

This goal is optional as the attacker may ignore it if an investigation is not a concern (i.e., if the attacker covers their traces or can guarantee that the transaction will go through before any flags are raised—for instance, when stealing amounts typical for the victim to transfer). Our attack satisfies this goal, as opposed to traditional adversarial attacks against ASVs, which do not produce samples in the victim's voice.

Knowledge. We are concerned with the most realistic black-box setting, where the attacker only has access to the VA's decision (accept/reject), without knowledge of the models' internals, training data or output probabilities.

Capabilities. *Access to a rooted phone (designated-app only):* The attacker can bypass the microphone on the device used to interact with the target to inject audio directly into the app. This requires that the attacker obtains *some* rooted device. We do *not* assume access to the victim's device. Rooting is possible as rootkits are available online [64]. Apps may be equipped with "root detectors", but these can often be evaded [44]. The ability to bypass the microphone has been assumed in many previous works [33]. This is not required for over-telephony attacks, as the attacker can even use a computer.

Access to audio recordings of the victim (designated-app & over-telephony): The attacker is in possession of sufficient data (and computational resources) to train SS/VC models in the victim's voice. Modern SS/VC systems accomplish this with only ∼15 minutes of victim speech [285]. Such data is easy to obtain thanks to social media and conference calls, especially for a determined attacker targeting a specific individual. Alternatively, zero-shot algorithms (which we do not consider) only require 5 seconds in the victim's voice (but only retain a 50% success rate against human listeners [261]).

Obtaining PINs (designated-app only): The attacker (pretending to be the victim) installs the app the victim uses to interact with the target on the attacker's rooted device. Most

banks allow their users to install their apps on multiple devices. They may require entering a PIN [19], which the attacker can obtain through social engineering [189]— a standard assumption [37]. In the over-telephony scenario, this requirement is waived, as the attacker may use any phone (even a landline) to place the call, and no app (therefore no PIN) is required.

3.4 Attacking Security-Critical VA Deployments

3.4.1 Attack Overview

Our attack is motivated by two main hypotheses:

H1: *There exist **identifiable**, **removable**, and **universal** key differences between human and machine speech.*

H2: *These nuances are among the **primary** telltales on which CMs rely to make their decision.*

Note: The term universal above implies that these differences (nuances) are agnostic to the user or spoken text.

Optimization-free universal attacks on ASVs are hard to realize as they require fathoming features that characterize the specific victim's voice and how they can be injected into different utterances. This is evident by the variety of model-specific, non-universal attacks against those systems (see §3.10), and absence of model-agnostic attacks, despite the recurrent emphasis on their importance [300]. However, based on ***H1***, which states that there are human-understandable universal differences (across users and speech samples) between machine and human speech, and ***H2***, which states that they are crucial for CMs to perform correct classification, we identify a new potential weakest link. Successfully eliminating those intuitive differences for a given spoofed sample through designated transformations can make the components protecting ASVs against spoofing (i.e., CMs) misclassify. This will leave the entire VA platform vulnerable, due to the known vulnerability of ASVs to spoofing attacks. Furthermore, this nature of the targeted nuances entails that defending against our attack will not be trivial, as we target core features for CMs, leaving them practically useless. The novelty in our work lies in the realization of these transformations that target commonalities of various CMs without resorting to transferability, which is of limited efficacy (see §3.6.1). The challenge is that these transformations need to be subtle to preserve the properties of the spoofed sample that make it effective against ASVs, and its textual content (necessary to bypass STTs).

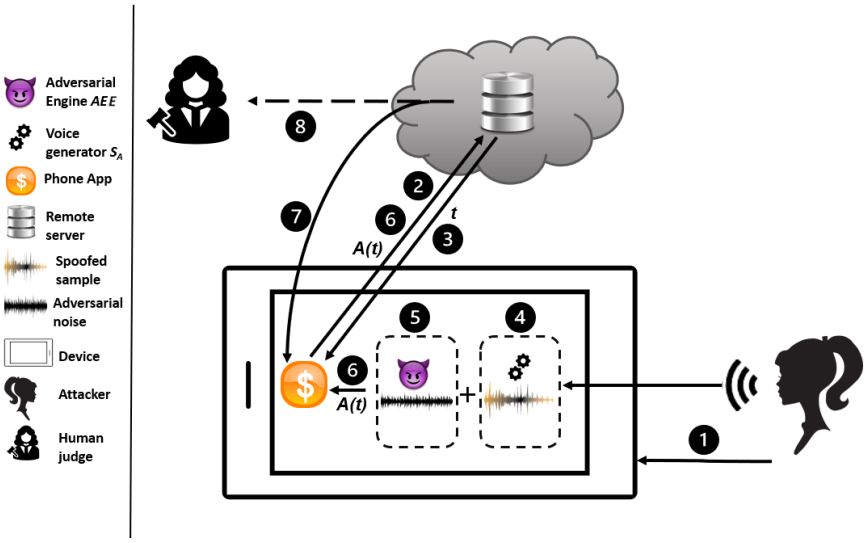


Figure 3.1: Attack Overview.

Overall, the attack is as follows (see Figure 3.1): 1) The attacker accesses the target-provided app (or places a call to the target) and issues a command triggering a transaction. 2) The target receives the request and 3) prompts the attacker with a random authentication phrase t. 4) The attacker generates the requested phrase in the victim's voice via an SS or VC algorithm S_A. 5) The result is forwarded to the adversarial examples engine AEE, which perturbs it using our transformations to generate a sample with identical content in terms of its text, while also preserving the voiceprint of the victim. 6) The adversarial example is injected into the app (call) and transferred to the target. 7) The target verifies that the sample bears the victim's voiceprint (ASV accepts due to the vulnerability of ASVs to spoofing attacks), the textual content of the sample is t (STT accepts), and the sample comes from a human (CM accepts). 8) If the transaction is later reported as suspicious, the human judge is fooled by the seemingly genuine sample and confirms the transaction's authenticity.

3.4.2 Universal Adversarial Transformations Against CMs

As explained in §2.3.1, adversarial attacks are typically crafted through gradient-based optimizations but it is also possible to generate adversarial examples through structured transformations. Here, we take the latter approach. We seek transformations than can eliminate identifiable and removable machine cues, while preserving quality. Hence, these

transformations must be crafted based on an understanding of the design of CMs and fake speech algorithms. We surveyed numerous papers to amass such an in-depth understanding, leading to the development and testing of various heuristics. Below we list the selected transformations:

F_1. *Replacement of Leading and Trailing Silence:* When humans speak into a microphone, there is static (white) noise from the background, in addition to involuntary sounds the speaker produces that accompany the speech (e.g., the speaker inhales before talking). Spoofed speech lacks these "non-speech" cues, which may influence the decision of CMs. In fact, applying voice activity detection (VAD) to filter out silences when training or evaluating CMs results in a performance degradation [161], proving these non-speech cues essential for CMs.

One may incorrectly assume that CMs can be trivially made ineffective by removing silence intervals, eliminating them as a deciding factor, and making CMs easier to fool [161]. However, when replicating these experiments, we find that eliminating silences makes CMs more likely to reject any input. Hence, this attack will not be successful (the attacker needs the model to accept spoofed samples) as the performance degradation when silence is stripped is the result of the inability of CMs to accept *bonafide* samples that lack the trimmed silence periods, which is of no use as spoofed samples are still rejected.

Nonetheless, "natural" silence can be copied from bonafide samples. F_1 replaces leading/trailing silence in spoofed samples with silence from benign samples. Appending longer silence intervals to increase the dependence on these cues is not a valid attack, as research shows trimming long silences possible, making them shorter but sufficient for classification [80].

F_2. *Elimination of Inter-word Redundant Silence:* F_2 further eliminates non-speech cues that identify spoofed speech by removing long "synthetic" silence intervals. SS/VC algorithms fail to generate statistically-natural silence to fill gaps between consecutive words. Spoofed silence intervals are often too quiet compared to human speech, since humans have air flow between their words. Similarly, natural speech exhibits an echo, as the different reflections of the same sound arrive at the microphone at different times (see F_4), populating those silent regions. Furthermore, spoofing algorithms often generate speech by dividing it into time steps, generating each independently. This step-by-step (often non-causal) procedure may cause the output to have extended unnatural silences, although the words at each step may have a (semi-)natural silence distribution between them. As opposed to leading/trailing silences, these cannot be replaced as their location mandates that the distribution of their static noise must be compliant with preceding and following intervals and replacing them with silence from different samples will not accomplish this. Thus, F_2

eliminates inter-word silences altogether. We find that since F_1 already injects human-like silences, in F_2 it is sufficient to remove redundant silent periods. Using the right parameters, we reduce only long (empty) silence intervals but do not eliminate them completely, so the output still contains the more lively (noisy) silences if they exist, sounding natural in terms of the speed at which words are uttered.

F_3. *Boosting the center of the spectrum:* The majority human speech energy is concentrated in lower parts of the spectrum. Thus, our hypothesis is that this frequency range naturally becomes the most critical in audio-related tasks compared to negligible higher frequency components. Furthermore, many audio classifiers (CMs, ASVs, ASRs) rely on tempo-spectral representations that emphasize details in lower frequencies while de-emphasizing high components (e.g., MFCC [223]) or have higher resolution at lower frequencies (e.g., CQCC [240]). CMs often use such front-ends. Moreover, the latest state-of-the-art CMs operate on raw waveforms. Due to the energy distribution of speech (concentration around lower frequencies), we expect these systems to naturally assign larger weights to lower components as well. Thus, CMs are more sensitive to cues in lower frequencies. Introducing perturbations manipulating the spectra s.t higher frequencies are amplified relative to these lower components decreases the magnitude of the cues that can be more precisely classified by CMs, while the regions that are less accurately learned will be made dominant. The lack of accuracy in this range causes misclassification.

This transformation amplifies the intensity of the signal's spectrum around the target region. The range $(1-4)KHz$ has the most considerable effect on speech intelligibility [82]. Higher frequencies (typically above 6KHz) are dominated by noise, less important to intelligibility, and are naturally of lower magnitudes. Thus, we ensure keeping these higher frequency components suppressed by using a sharp (fast-decaying) filter around the center of the spectrum.

F_4. *Local Echo:* This is an extension to F_2. As opposed to spoofed speech, natural speech is typically recorded with a microphone, which receives its input from a user speaking into it. The user's voice does not travel to the microphone in a single trajectory. Instead, voice from their mouth travels in many directions and gets reflected at different angles and times, echoed back to the microphone. With time, the amplitudes of the reflected particles belonging to each utterance dissipate, no longer being recorded. The lack of echo in spoofed speech can serve as a sign telling benign and spoofed samples apart. F_4 generates local echo, introducing 1ms-shifted copies to the signal over a short period (100ms), decreasing the amplitude of the repeated copy as we advance in time.

F_5. *Pre-emphasis:* This transformation applies pre-emphasis to the input signal. Pre-emphasis [113] is used to alter the signal at different frequencies by amplifying the relative

magnitudes of higher frequency components at the expense of lower frequencies. It is popular in high-speed transmission systems, creating noise-robust signals before transmission. Upon receipt of the signal, the operation is inverted via de-emphasis.

The intuition behind using pre-emphasis is similar to that motivating F_3— the uncertainty of CMs at higher frequencies. Yet, a drawback of pre-emphasis is that its frequency response is monotonically increasing, leading to large magnitudes at very high frequencies, and yielding outputs with flat spectra (the energy is evenly distributed along the frequency axis). Those are unnatural compared to normal speech, wherein the energy is concentrated at lower frequencies, and will sound unrealistic. Instead, as in F_3, the target region we wish to amplify is the center of the spectrum (around 1KHz-6KHz).

To circumvent this, we use a smaller coefficient for the operation (0.5) instead of typical large coefficients (0.97) to ensure the output signal complies with the spectral map of natural speech. This limits the usability of F_5, although it is still highly effective. F_3 can, hence, further boost the attack's performance as it enables defining the specific region to amplify, leaving out undesired frequencies. Yet, F_5 enables us to systematically manipulate the entire spectrum as a well-established technique, preserving inter-connections between different frequency components, which is harder with F_3.

F_6. *Noise Reduction:* The samples produced by spoofing algorithms often exhibit machine noise, which characterizes the algorithms used to produce them. In a way, this noise can be thought of as a fingerprint attesting to the algorithm's imperfections. As opposed to naturally-occurring (additive) noise, it does not exhibit the same normal distribution over the spectrum, and will be centered around specific frequencies, forming a cue that CMs may spot. This computational limitation is most likely due to dataset biases, as the settings under which the training data is gathered (e.g. recording devices) may introduce such artifacts. If eliminated, CMs would lose the ability to rely on them to output a decision.

We implement F_6 via spectral gating [216], which operates by first calculating the tempo-spectral representation of the signal. Afterward, for each frequency, the mean and standard deviation over time are calculated. Lastly, at each time step, if the intensity at some frequency is not sufficiently larger than the mean for that frequency over time, this component is eliminated. This filters out continuous background noise at each frequency, leaving only unique, audible (non-noise) information. Additionally, we find introducing additive (white) noise at this layer replaces the mechanic one with the natural counterpart found in human samples and enhances the attack.

F_7. *Adversarial speaker regularization (ADVSR):* We include a final optimization-based layer ($ADVSR$). However, this does not change the optimization-free, model-agnostic nature of our attack, since we do not use a shadow CM to perform the optimization. The

optimization is done via a model designed for an adjacent task—an ASV. The objective of this step is not to achieve high success against target ASVs, nor do we rely on transferability. The goal is to utilize a shadow ASV to optimize the spoofed sample to increase its ability to fool CMs. Hence, this step is still considered a model-agnostic transformation. ASV-based optimization can eliminate machine artifacts. The idea is to "engrave" the user's voiceprint into the spoofed sample. The rationale is that, given a reference sample $\boldsymbol{y}$ by the victim user, and the spoofed sample $\boldsymbol{x}$, a non-zero ASV loss (for $\boldsymbol{x}$ w.r.t. $\boldsymbol{y}$), indicates that there is some uncertainty as to whether the speaker in both samples is the same, and we attribute that to one of the samples being "spoken" by the machine. When the ASV removes these differences, machine cues are further eliminated as a byproduct, making the sample even less machine-like and capable of bypassing the CM.

Aside from $F_1 - F_7$, we experimented with various other techniques (excluded for limited success), including CM-based adversarial attacks (see §3.6.1), vocal tract copying [48], and marginal filtering at fundamental frequencies and harmonics. We leave exploring additional transformations to future work.

3.4.3 Attack Implementation

First, we replace machine silence with "realistic" silence segments. We use *Librosa* [170] to trim leading and trailing silence, with a threshold of -25.0 db, and replace it with silence from a bonafide random sample of the same speaker (F_1). F_2, uses $WebRTCVad$ [263] to remove inter-word extended silences (aggressiveness=3), with a frame of $30ms$. We find that this algorithm and hyperparameters retain necessary silence intervals and do not eliminate them completely, satisfying the requirements of F_2. We preserve the original sample length after steps F_1 and F_2 by choosing bonafide sample periods that match in length those that F_1 and F_2 trim.

Afterward, we apply F_3 by computing the FFT of the input and amplifying the frequency range 1-6KHz by 1.5 using a sharp Butterworth BPF (with an order of 20). This limit of 1.5 ensures that the outputs do not sound suspicious to humans and avoids violating the spectral map of normal speech (see §3.4.2). F_4 is introduced by adding shifted copies (by $1ms$ at a time) with decreasing amplitudes. Assuming the original amplitude is A, the amplitude of the i'th shifted copy is $A/(8 * i + 1)$, since the longer it takes the echo to reach the microphone, the more energy it loses. Next, we apply F_5 based on *Librosa*'s implementation with a coefficient of 0.5. In F_6, noise reduction is applied [216] and additive noise is introduced instead with a limited amplitude of $\epsilon = 0.0035$ (to avoid generating noisy samples). To preserve the effects of F_1, we apply noise reduction between the leading

and trailing silences only. Finally, we perform F_7 with the ASV described in §3.5. Since the ASV's loss requires a reference voiceprint of the victim, we use multiple bonafide (victim) recordings. We generate the adversarials using *I-FGSM* [140] and $\epsilon = 0.0015$.

The transformation order and parameters were determined empirically and can be explained as follows: F_1, F_2 are order-invariant, operating on different regions. F_3, F_5 are order-invariant; both perform (frequency) *amplification* operations. F_1, F_2 replace signal fragments and should precede F_3, F_5 to avoid discarding some of their effects. F_6 introduces noise that must be minimal (inaudible). Hence, it is applied after amplification (F_3, F_5). F_7 must be last since it is a volatile voiceprint-based transformation, while others are voiceprint-agnostic (destructive to F_7 if applied afterward). F_4's (echo) placement (determined empirically) must be between F_2 and F_7.

3.5 Experimental Setup

This section presents the models used, their baseline performance, and our datasets.

System specifications. All adversarial examples were generated on a server running Ubuntu 20.04 on a PPC64LE, POWER8NVL 2.33 GHz CPU with 16 physical (128 virtual) cores and 1TB RAM and 4 Tesla P100-SXM2 GPUs with 16G of memory each (although our experiments used one GPU at a time). To mount our over-telephony attack, we set up a dummy server as an Amazon Connect [12] instance and used Twilio [244] to programmatically place calls to it over which the samples were delivered. The samples were recorded at the receiver in AWS, downloaded to our server, and fed into the CMs.

Sampling & encoding. All samples in our datasets are sampled with a rate of 16KHz and represented as FP arrays in the range **[-1.0, 1.0]**— a standard representation (we normalize our transformations' outputs to this range). When transmitting samples over the phone, due to bandwidth limitations of the carrier (Twilio), they are received at a sampling rate of 8KHz, which we then up-sample to 16KHz to feed into our targets.

Systems. We select top-performing models from the literature and industry, depending on availability.

CMs. Commercial CMs, such as Nuance's, are proprietary and closed-sourced, without public APIs. We reached out to leading platforms, such as Aculab and ID R&D Voice, but never heard back. Thus, to evaluate our attack against the state-of-the-art, we use open-source systems from the ASVspoof2019 challenge [241]. We focus on these since the challenge has long been the platform where novel CMs emerge. The challenge received contributions from

renowned corporations (e.g., Google, which contributed to the dataset [262]), demonstrating its importance by industry standards. We do not restrict ourselves to systems that actively participated in the challenge and include more advanced CMs that have since appeared with superior performance (w.r.t. the challenge's task). We consider the best reported models [123, 234] (released 2021), and for which the authors provide pre-trained models and results (that we verified)—see Table 3.1. We use pre-trained models to avoid introducing any errors while reproducing them. These models' performance is on par with leading commercial systems. Specifically, the world-leading ID R&D Voice [114] [191] was the winner of the ASVspoof2019 challenge, achieving an equal error rate (EER) of 0.22%. The also proprietary system ranking second achieved an EER of 1.86% [241]. Our chosen systems achieve similar numbers, with some ranking between the top two contestants.

Since the list in Table 3.1 was compiled, new systems have emerged. Therefore we include a representative of such potentially more robust models. We use the state-of-the-art WAV2VEC [235], a model designed for the ASVspoof2021 challenge, whose objective was to build generalizable CMs robust to the audio channel effect (can correctly classify samples even over phone calls). The model was evaluated on the ASVspoof2021 evaluation dataset, which augments the ASVspoof2019 evaluation subset with samples transmitted over the phone. This model is capable of operating on clean audio (as found in the ASVspoof2019 dataset) and phone audio and is so far the best at performing both tasks without needing further calibration [235]. Note that WAV2VEC achieves the lowest EER (first row in Table 3.1), and that this EER is w.r.t. the more challenging, augmented ASVspoof2021 dataset. Since WAV2VEC is a very recent model, we only include it in our final experiments that evaluate our full attack.

System	Front-end	EER
WAV2VEC [235]	Raw-audio	0.82%*
AASIST [123]	Raw-audio	0.83%
AASIST-L [123]	Raw-audio	0.99%
RAWGAT_ST (mul) [234]	Raw-audio	1.06%
SSNet [111]	Raw-audio	1.64%
RAWDARTS [84]	Raw-audio	1.77%
MCG [154]	CQT	1.78%
LCNN_LSTM [255]	LFCC	1.92%
MLCG [154]	CQT	2.15%
AIR [294]	LFCC	2.19%
Res2Net [153]	CQT	2.50%
AIR_AM [294]	LFCC	3.26%
DARTS [83]	LFCC	4.96%
LCNN [146]	LPMS	5.06%

Table 3.1: Top-performing CMs on the ASVspoof2019 LA evaluation dataset with public code and pre-trained models as of October 2021. *In the first row, WAV2VEC is the state-of-the-art CM and its EER is w.r.t. the more comprehensive ASVspoof2021 LA evaluation set.

Due to the known possible lack of generalization to the over-telephony setting [277], some CMs may not be appropriate for this setting. This does not apply to WAV2VEC,

which was specifically designed and evaluated for this task. Thus, in the over-telephony experiment, we naturally include WAV2VEC. For other CMs, we determined which can generalize to this setting and calibrated their decision thresholds accordingly. Details are in Appendix §A.2. In conclusion, we only found seven other models to be somewhat generalizable, although their performance is inferior to WAV2VEC.

Our chosen CMs vary significantly in their front-ends (see Table 3.1) and architectures, which proves our attack bypasses any VA platform.

ASVs. We use various architectures to represent the state-of-the-art (GMM i-vectors, x-vectors, and end-to-end). Using KALDI [204], we design three GMM i-vector models with different dimensionalities and front-ends (ASVShadow, ASVTarget1, ASVTarget2), and a single x-vector model (ASVTarget3). KALDI is widely employed in academic and industrial projects and was used to evaluate previous work [283]. Additionally, we use Resemblyzer [208], which implements Google's GE2E [248] as an end-to-end ASV. Resemblyzer is popular and has been used in academic work [261]. Finally, we include the commercial Amazon Connect Voice ID [13], which is available for call centers using Amazon Connect [12] to verify callers' identities. The ID delivered in each call was used to invoke Voice ID with the (previously-uploaded) voiceprint of the speaker to whom the delivered sample is claimed to belong. Each voiceprint consists of 30 seconds of speech, obtained by concatenating bonafide samples from the ASVspoof2019 evaluation set for each speaker. At the time these experiments were conducted, Voice ID was not using a CM. Details of all ASVs are in Appendix §A.1. We used ASVShadow for F_7. All other models were used as targets. The difference in architectures and training data preserves blackbox assumptions.

STTs: We use two common commercial (blackbox) STTs: Google Cloud STT [89] and Microsoft Azure STT [174].

Training datasets. *CMs:* We mainly experiment with pre-trained CMs developed for the ASVspoof2019 challenge, which (except WAV2VEC) were trained on the training subset of the LA portion of the ASVspoof2019 dataset [256]. It includes samples from 107 speakers (46 males/61 females). For each speaker, there are bonafide and spoofed (VC/SS) samples. The dataset is partitioned into 3 sections: training, development, and evaluation. The training and development datasets contain samples from 20 speakers each, while the evaluation samples come from 67 speakers. The subsets are disjoint in speakers and spoofing algorithms. For WAV2VEC, the pretrained model uses the ASVspoof2019 training subset, augmented with additive and convolutional noise.

ASVs: Training the GMM i-vector ASVs was done using the bonafide samples from the (disjoint) development and training subsets of ASVspoof2019 for the shadow and targets, respectively. The x-vector system was trained using Voxceleb1 [179], augmented using

MUSAN [226]. Resemblyzer and AWS Voice ID come pre-trained. To use them, their thresholds must be calibrated. We generated calibration data from the evaluation subset of ASVspoof2019 (see Appendix §A.1 for details).

STTs: We used pretrained API-accessible systems.

Evaluation dataset. Spoofed samples in the ASVspoof2019 evaluation subset were generated by various state-of-the-art spoofing algorithms (as of 2019). As the output quality of SS/VC models depends on the availability of training samples and capabilities of the algorithms, we find samples in this subset to be of different qualities. Since our attack optimizes high-quality spoofed speech (such as [223]), we decided to focus on a reduced set of this subset, including only samples of the expected quality that can initially bypass STTs and fool humans. This decision is further motivated by the need to have a reduced set to evaluate in our large-scale experiments (especially in the costly and time-consuming over-telephony setting). After listening to a few thousands of recordings, we chose 474 spoofed samples belonging to 48 users, generated by 13 different algorithms we deemed of satisfactory quality. Compared to previous works considering over-telephony adversarials [1] and spoofed speech [25], this number is similar and the variety in users and algorithms is more extensive. To promote future research, we make our high-quality reduced subset available[2]. We also include the spoofed and adversarial samples we use in the user study in §3.8.

Evaluation metrics. We define our attack's success as the Acceptance Rate (AR) for each system type. For CMs, this is the probability that the sample is deemed bonafide. For ASVs, this is the probability that given a reference victim voiceprint, the ASV's comparison of the attack sample against this reference leads to acceptance. For STTs, acceptance indicates a successful comparison of the transcribed text from the sample against the phrase expected by the system. We assume spoofed samples always satisfy this property, due to the quality of modern audio spoofing. For adversarial examples, acceptance amounts to the transcribed texts from the original sample and the adversarial generated from it matching.

3.6 Attacking Designated-App Authentication

This section focuses on the designated-app scenario (see §3.3.2), where authentication phrases are passed to the server as waveforms over TCP (i.e., reliably, without packet loss, jitter or lossy encoders [185]). First, we show results against each of the CMs, ASVs, and STTs independently. Then, we present the joint results for them combined. Recall that we

[2]Available at the GitHub repository accompanying this work

Frontend	LPMS	LFCC				CQT			Raw Waveform				
System / *Transformation*	LCNN	AIR	AIR_AM	DARTS	LCNN_LSTM	MCG	MLCG	RES2NET	SSNET	RAWGAT_ST	RAWDARTS	AASIST	AASIST-L
None	3.16%	2.32%	3.59%	4.01%	2.95%	1.90%	4.01%	2.95%	2.74%	3.16%	1.90%	2.32%	2.32%
F1	13.71%	15.61%	13.92%	13.08%	10.97%	12.66%	9.28%	4.85%	12.66%	7.59%	1.69%	4.85%	5.27%
F1-F2	19.41%	23.00%	17.30%	16.03%	15.40%	16.24%	15.82%	4.22%	41.56%	9.70%	3.38%	13.29%	10.97%
F1-F3	20.25%	38.40%	28.90%	14.98%	24.89%	32.70%	20.04%	7.59%	43.46%	9.92%	4.64%	15.40%	13.08%
F1-F4	19.83%	39.87%	30.17%	16.46%	24.89%	34.60%	20.46%	8.44%	47.05%	10.76%	5.49%	15.82%	14.35%
F1-F5	**21.31%**	44.30%	34.39%	20.46%	24.05%	45.99%	27.85%	8.02%	54.01%	11.81%	12.03%	18.99%	17.93%
F1-F6	12.03%	45.99%	42.19%	**20.68%**	**25.11%**	**67.51%**	45.15%	**16.03%**	57.81%	11.18%	**15.82%**	24.68%	21.73%
F1-F7 (Full Attack)	17.93%	**58.02%**	**51.27%**	20.04%	15.61%	65.82%	**52.74%**	15.82%	**61.81%**	**12.03%**	**15.82%**	**24.89%**	**21.94%**

Table 3.2: Attack results against CMs. Bold entries represent the best success results for each system.

target CMs—success against STTs/ASVs is due to the quality of the used spoofed speech, the vulnerability of ASVs to it, and our attacks' ability to preserve these properties.

3.6.1 Results Against CMs

Shattering the myth of transferability: The reader may question the need for our transformations given the existence of conventional adversarial techniques. Assuming a query-limited attacker leaves us with transferability attacks (crafted against a shadow CM and transferred to the target), as done in previous works [160, 295, 293, 292]. However, transferability was proven ineffective for other audio-related tasks [3]. We prove that it is not a viable attack against CMs either, pinpointing flaws in previous works. Details are in Appendix §A.3.

Our transformations: The collective (cumulative) attack results are in Table 3.2. The transformations should be performed collectively as they are motivated by various hypotheses and target different cues. We report individual evaluations of each transformation in Appendix §A.4. In a nutshell, each transformation individually incurs a significant increase in the attack's success against several (but not necessarily all) CMs compared to spoofing baselines. Nonetheless, these do not affect all CMs similarly as they employ various heuristics, leading to a minor degradation w.r.t. the baselines on rare occasions. The ensemble attack (discussed below) remains universal, far outperforming singletons.

For F_1 (silence replacement), compared to spoofed samples alone without any additional transformations (first row in Table 3.2), it is clear that F_1 leads to a considerable degradation in the performance of several CMs. Even CMs that were completely unaffected by additive noise or transferability (RAWGAT_ST, AASIST, AASIST-L—see Tables A.3 & A.4 in

Appendix §A.3) exhibit a performance degradation under this simple transformation. This is evidence that silence intervals of bonafide recordings encode non-speech liveness cues missing in machine speech. These non-speech cues can be injected by replacing silent intervals with genuine silences, as done by F_1.

Next, we study F_2 (silence elimination). As explained in §3.4.2, additional non-speech cues guiding CMs are embedded in inter-word silence intervals, as SS/VC algorithms often fail to mimic genuine silences, generating long, mute segments. This is cemented by F_2's findings, demonstrating a significant increase in the attack's success against most CMs.

F_3 (spectrum center boosting) is also highly effective. We can see that increasing the relative amplitudes of the uncertainty regions at the expense of lower frequencies (see §3.4.2) drastically increases the success rates in general.

The results for F_4 (local echo) indicate that the contribution of this transformation is only incremental compared to the other transformations. Nonetheless, we still see some improvement over the cumulative attack without F_4 for the majority of CMs, especially for the harder systems to spoof, such as RAWGAT_ST and RAWDARTS. These systems have proven so far to be far more robust than counterparts and therefore any improvement is significant and proves the reliance on this echo as a cue. For instance, this is the first time we see a success rate of above 5% against RAWDARTS.

F_5 (pre-emphasis) is among the leading transformations, especially against CMs operating on raw waveforms. Recall that for the pre-emphasis filter, to ensure the output sample's spectrum conforms to the spectral map of natural speech (see §3.4.2), we chose a smaller constant than the typical. Interestingly, while large coefficients produce unnatural-looking (sounding) spectra failing human inspection, most CMs (except $LCNN$, $LCNN_LSTM$, $DARTS$) are vulnerable to pre-emphasis with a large coefficient (0.97) (see Appendix §A.4). Not only will such samples likely fail human inspection, pre-emphasis with large coefficients can also be defeated by adding a component measuring the statistical distribution of the signal's spectrum to reject unnatural-looking samples. Comparing the spectral density against the natural distribution of speech is a vital step currently missing in many CMs. Nonetheless, our attack produces natural-looking examples, robust to this proposed defense.

F_6 (noise reduction) eliminates characteristic noise due to imperfections of spoofing algorithms that may be identified by CMs, replacing it with additive noise following the observation that bonafide samples often exhibit such noise. We see a significant increase in the success against most CMs (except $RAWGAT_ST$), making F_6 among the most important transforms, and highlighting CMs' dependence on *removable* cues.

The last layer is ADVSR (F_7). When coupled with the rest of the transformations, it

achieves the highest attack success rate for multiple CMs, cementing the reasoning behind it.

State-of-the-art (WAV2VEC): WAV2VEC was designed for channel generalization and trained on noise-augmented data, which was proven to increase noise robustness and improve generalization [235]. This is a natural defense against our attack, since the non-speech cues we target in F_1 and F_2 are linked to the absence of natural silence in spoofed samples. Additionally, our transformations boost central frequency components (F_3 and F_5). Although we amplify these components that are of utmost import to intelligibility and affect CM classification, higher frequencies are amplified as a byproduct, which may also contribute to our attack's success. Higher components include noise-dominated information, which CMs should overlook (see §3.4.2). WAV2VEC was designed to operate on telephony data, which often lacks components above 4KHz (see §3.7). Hence, it should be able to disregard these components, while other CMs may be vulnerable to perturbations in this range.

The spoofed samples have an initial success rate of 1.89% against WAV2VEC, making it the most robust CM we experiment with. After applying our transformations, the success rate increases to 11.6% (a 6.14x performance degradation), making it highly untrustworthy by the standards of a strict security platform. In conclusion, even the state-of-the-art is vulnerable to our transformations. The additive and convolutional noise used to train WAV2VEC does not necessarily capture microphone or articulatory sounds, which are the liveness cues found in the silence of bonafide speech, and therefore is less effective, leaving suitable data augmentation an open challenge. Furthermore, WAV2VEC shares the vulnerability of being more accurate at lower (compared to central) frequency regions. Resistance in higher frequencies is found to be of limited efficacy.

Summary: Our attack defeats state-of-the-art CMs, with a success rate of 11.6% against the most robust CM (WAV2VEC) and 65.82% against the most vulnerable (MCG). The main takeaway is that the cues these systems rely on can be easily spoofed. We also proved that adversarial transferability across CMs is an impractical attack strategy (see Appendix §A.3).

3.6.2 Results Against ASVs

Table 3.3 shows the success rates against ASVs. The first row shows the success of spoofed samples. Currently, there are even stronger spoofing algorithms that achieve higher success rates against ASVs. For instance, Wenger et al. [261] achieve a success rate of 100% against Resemblyzer, whereas our spoofed samples only attain a rate of 85.65%. Nonetheless, our included spoofed samples demonstrate the vulnerability of ASVs to spoofing and the need

System / *Attack*	*ASVShadow*	*ASVTarget1*	*ASVTarget2*	*ASVTarget3*	*Google/Resemblyzer*	*AWS Voice ID*
Spoofing	65.19%	66.24%	59.92%	99.79%	85.65%	69.15%
Our attack	100%	94.3%	81.86%	100%	75.1%	65.3%

Table 3.3: Attack results against ASVs.

for CMs. Our attack's success (second row) proves it retains the victim's voiceprint in the spoofed samples as it does not lead to considerable degradation against any of the ASVs. The systems belonging to entirely different architectures (ASVTarget3, Resemblyzer) compared to our shadow (ASVShadow), in addition to AWS Voice ID, which the results imply is of a different architecture, are only slightly affected by our transformations, maintaining high success. ASVTarget1 and ASVTarget2 exhibit a significant increase in the attack's success. We attribute that to our ASV-based adversarial layer (ADVSR), where ASVShadow is used to apply the optimization. Since all i-vector models share the same architecture (up to the variation in dimensionality), adversarial perturbations naturally transfer to the other two i-vector models (ASVTarget1/2) as opposed to different architectures. These results align with previous findings regarding the lack of transferability across different ASVs/ASRs [37].

We reported our findings about Amazon Connect Voice ID to Amazon, which has since deployed a CM. Evaluating our transformations against Voice ID's CM is left to future work.

3.6.3 Results Against STTs

To evaluate our attack w.r.t. STTs, we use Google Cloud's [89] and Microsoft Azure's STT modules [174], and test whether 25 random adversarials generated using samples from our evaluation dataset are transcribed to the same content as the original samples. The success rates are 76% and 96% against Google and Azure respectively, (see Appendix §A.5).

3.6.4 Results Against the Combined Defense

We evaluate success against the combined defense components (CMs, ASVs and STTs). The human judge is only consulted after the fact in special cases (see §3.3). Thus, we exclude this component here. This is the most restrictive (and practical) setting under which VA attacks can be evaluated. We report success rates for 3 and 6 authentication attempts since these are typical numbers used in practice (see §3.3).

CM \ ASV	*ASVTarget1*		*ASVTarget2*		*ASVTarget3*		*Resemblyzer*		*AWS Voice ID*	
	Spoofing (3 Atts./6Atts.)	*Our Attack (3 Atts./6 Atts.)*	*Spoofing (3 Atts./6Atts.)*	*Our Attack (3 Atts./6 Atts.)*	*Spoofing (3 Atts./6Atts.)*	*Our Attack (3 Atts./6 Atts.)*	*Spoofing (3 Atts./6Atts.)*	*Our Attack (3 Atts./6 Atts.)*	*Spoofing (3 Atts./6Atts.)*	*Our Attack (3 Atts./6 Atts.)*
LCNN	1.88/3.72	**34.7/57.36**	3.73/7.33	**31.79/53.47**	9.18/17.52	**39.49/63.39**	2.5/4.94	**17.38/31.74**	1.91/3.78	**4.9/9.55**
AIR	1.88/3.72	**84.79/97.69**	3.12/6.14	**78.11/95.21**	6.8/13.14	**87.42/98.42**	2.5/4.94	**71.15/91.68**	1.91/3.78	**60.9/84.71**
AIR_AM	2.5/4.94	**79.27/95.7**	4.35/8.5	**71.39/91.82**	10.36/19.65	**82.52/96.94**	3.73/7.33	**62.29/85.78**	0.63/1.25	**48.11/73.07**
DARTS	1.25/2.49	**39.1/62.92**	3.12/6.14	**35.92/58.93**	10.94/20.69	**43.3/67.86**	1.88/3.72	**23.9/42.09**	1.91/3.78	**6.51/12.6**
LCNN_LSTM	1.25/2.49	**32.62/54.6**	1.88/3.72	**27.92/48.04**	8.59/16.44	**35.11/57.89**	2.5/4.94	**20.21/36.34**	1.91/3.78	**7.57/14.56**
MCG	0.0/0.0	**90.07/99.01**	1.25/2.49	**84.16/97.49**	5.56/10.82	**91.83/99.33**	1.88/3.72	**80.21/96.08**	1.91/3.78	**71.63/91.95**
MLCG	4.96/9.67	**80.4/96.16**	5.56/10.82	**73.24/92.84**	11.53/21.72	**83.68/97.34**	10.36/19.65	**70.19/91.11**	9.38/17.88	**59.69/83.75**
RES2NET	2.5/4.94	**30.94/52.31**	3.73/7.33	**28.78/49.28**	8.59/16.44	**35.51/58.42**	4.96/9.67	**22.08/39.29**	3.18/6.25	**11.21/21.17**
SSNET	4.35/8.5	**88.22/98.61**	4.35/8.5	**82.52/96.94**	8.0/15.35	**89.72/98.94**	5.56/10.82	**80.58/96.23**	4.43/8.67	**78.83/95.52**
RAWGAT_ST	3.73/7.33	**24.8/43.45**	4.35/8.5	**19.27/34.83**	9.18/17.52	**27.92/48.04**	7.4/14.25	**16.9/30.95**	0.63/1.25	**9.15/17.46**
RAWDARTS	0.63/1.25	**33.04/55.16**	1.25/2.49	**30.09/51.13**	4.96/9.67	**35.51/58.42**	2.5/4.94	**19.27/34.83**	2.56/5.05	**8.1/15.55**
AASSIST	4.35/8.5	**48.36/73.33**	4.35/8.5	**41.03/65.23**	6.8/13.14	**51.45/76.43**	4.96/9.67	**40.65/64.77**	2.56/5.05	**30.21/51.3**
AASSIST-L	3.12/6.14	**43.67/68.27**	1.88/3.72	**35.92/58.93**	6.8/13.14	**46.6/71.48**	4.96/9.67	**35.92/58.93**	1.28/2.55	**26.66/46.22**
WAV2VEC	4.35/8.5	**21.62/38.57**	3.12/6.14	**17.86/32.53**	5.56/10.82	**24.35/42.78**	4.96/9.67	**15.44/28.49**	4.43/8.67	**8.62/16.49**

Table 3.4: Attack results (%) against combined ASV-CM-STT deployments.

The results were calculated as follows: for each ASV-CM pair, we extracted the number of samples that bypassed the two systems jointly and divided it by the size of our evaluation dataset (474), yielding $\mathcal{P}_{\mathcal{AC}}$—the probability to bypass the ASV-CM pair. Given that our experiment with STTs only included 25 samples, we could not calculate the joint probability against all systems similarly. We assume the success rates against STTs to be independent of the success rates against the other components (since our attack is almost always successful against STTs)—in the future a more precise evaluation can be done. We use the average success rate against the two STTs (86%). Thus, the success rate against an ASV-CM-STT deployment becomes $\mathcal{P}_{\mathcal{ACS}} = \mathcal{P}_{\mathcal{AC}} * 0.86$. Finally, even though our transformations lack randomness (except ADVSR), we still assume that success at each attempt is independent of other attempts, due to the large space of spoofing algorithms and the infinite space of recordings of the victim that can be used to train them. Thus, success after n attempts has the probability $1 - (1 - \mathcal{P}_{\mathcal{ACS}})^n$.

The results are in Table 3.4. The combined systems indeed provide strong security against spoofing, with success rates as low as 0% even after 6 attempts (MCG/ASVTarget1). Yet, all combinations become unreliable against our attack, with the strongest (LCNN/Voice ID) suffering success of 9.55% after six attempts. For others, such as MCG/ASVTarget1, we approach 100% success. These results demonstrate how the strictest VA can be bypassed.

Frontend	*CQT*		*Raw Waveform*					
System / *Attack*	*MCG*	*RES2NET*	*SSNET*	*RAWGAT_ST*	*RAWDARTS*	*AASIST*	*AASIST-L*	*WAV2VEC*
Spoofing	10.95%	10.52%	13.8%	8.09%	11.55%	8.67%	10.53%	1.52%
Our attack	32.46%	30.70%	57.46%	22.15%	23.03%	26.54%	23.03%	6.03%

Table 3.5: Attack results against over-telephony-network CMs.

3.7 Over-Telephony-Network Attacks

To the best of our knowledge, we are the first to explore the over-telephony setting for *targeted* adversarial attacks—the only known adversarial attack against VA over the phone is non-targeted [1]. We present superior attacks impersonating specific victims and fooling security-critical VA deployments. Adversarial attacks over the phone are harder due to three main challenges: transcoding, packet loss, and jitter [3]. An additional factor is bandwidth: Samples are typically recorded with off-the-shelf devices at a high sampling rate—for instance, our inputs from ASVspoof2019 were recorded at 16KHz. However, phone carriers typically operate at a lower rate (8KHz), causing information loss at high frequencies. Our goal here is to ensure that our attack can withstand these distortions.

We used the setup in §3.5 to send samples over the phone. While Amazon Connect is robust to packet loss and jitter, the transcoding and bandwidth losses persist. As explained in §3.5, only WAV2VEC and seven other CMs are effective in the over-telephony setting and only they are considered. Table 3.5 shows the results of our over-telephony experiments involving the same 474 samples from §3.6. The results clearly demonstrate the threats associated with our attack even in the over-telephony setting. Even the most powerful CM for the over-telephony task to date (WAV2VEC) is affected, bringing the success against it to 6.03% from 1.52% (3.97x). While the attack is less successful than in the app setting, it is still highly powerful especially after multiple attempts.

3.8 Fooling a Human Judge

In our threat model, attacks must fool humans (HJ) if they are asked to inspect the samples (see 3.3). Given an audio sample and a claimed user identity (to whom the sample supposedly belongs), HJ verifies that the sample comes from that user. Since HJ is not necessarily acquainted with the user, this is done via a comparison to samples previously

provided by the user (e.g., at enrollment). We restrict the spoofed samples involved in this study to those belonging to the reduced set of 474 (high quality) samples introduced in §3.5. The adversarial examples were generated from spoofed samples in this set using our *Full attack* algorithm. The study also includes bonafide samples, selected from the evaluation subset of ASVspoof2019. We do not listen to adversarial samples before using them in our study. We use the same human-imperceptibility study from FakeBob [37], randomly assigning participants one of two tasks to be completed online:

Task 1: Clean or Noisy. Participants listen to 24 samples and indicate whether each is noisy. The options are *clean*, *noisy*, or *not sure*. We randomly select 12 spoofed samples from the batch described above and 12 adversarials, crafted from spoofed recordings in the same batch (not necessarily the same samples). To ensure quality of the responses, we include three additional samples that are completely silent as a concentration test. Respondents who find these samples noisy fail the concentration test and their questionnaires are excluded.

Task 2: Identify The Speaker. Participants listen to 24 sample pairs and indicate for each pair whether the samples were uttered by the same speaker. The options are *same*, *different*, and *not sure*. We randomly select 6 speakers (3 males and 3 females) from the reduced set. For each speaker we generate four pairs: 1) *Same Benign-Benign (SBB)*—bonafide samples by the same speaker, 2) *Different Benign-Benign (DBB)*—bonafide samples from different (same gender) speakers, 3) *Same Benign-Spoofed (SBS)*—samples by the same speaker; one bonafide and one spoofed, 4) *Same Adversarial-Benign (SAB)*—samples by the same speaker; one bonafide and one adversarial. As a concentration test, we include three opposite-gender pairs of bonafide samples. Responses with answers to the concentration tasks not finding the speakers different are disqualified (as in FakeBob). To isolate the effects of our attack, adversarials in SAB pairs were all generated from the spoofed recordings in the SBS pairs. Our objective is comparing the adversarials' results to those collected for their spoofed precursors. We include the SBB and DBB pairs as a quality test (see below). Compared to FakeBob, our task adds two categories, SBS and SAB, and omits bonafide-adversarial pairs, since they generate adversarials from benign samples while we use spoofed samples.

For both tasks, we only select adversarials that successfully bypass several CMs to evaluate the attack's overall potential.

Results. We ran our IRB-approved study on MTurk [11] and received 21 responses per task. We restricted our participants to those with an MTurk approval rate of 95% or higher. Out of all responses, two (one per task) failed the concentration test, leaving us with 20 responses for each. As opposed to FakeBob, we reveal our study's purpose to participants, guaranteeing they simulate real judges inspecting suspicious samples.

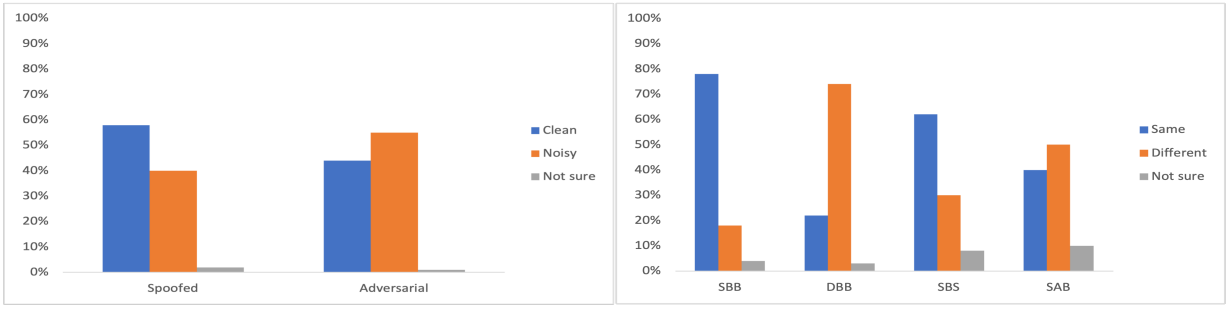


Figure 3.2: *Results for Tasks 1 and 2.*

Task 1. As shown in Fig. 3.2, 40% of our participants heard noise in spoofed samples, compared to 55% for adversarial samples. Spoofed samples are machine-generated, which may leave cues that can be spotted by a vigilant listener. This explains the relatively high rate of spoofed (baseline) samples flagged as noisy. Modern algorithms can generate better fake speech and reduce these rates [285]. However, our transformations increase this rate by only a factor of 1.375x (from 40% to 55%), which is relatively low. Compared to FakeBob, our transformations are much less obvious as their attack produced adversarial samples ∼3x noisier on average w.r.t. their initial samples. We focus on this degradation factor, since their attack starts from cleaner (bonafide) samples, while ours modifies fake speech, which may be initially noisier. Finally, noise level is simply a quality metric, not critical for determining the attack's potential in fooling humans (even natural samples may include noise). *Task 2* is the more relevant metric.

Task 2. As done in FakeBob and in accordance with our description of HJ's role, we say that a listener can differentiate between the speakers of two samples only if the user selects "different" for the two samples in the pair ("not sure" still indicates uncertainty). 74% of our participants believe that samples in DBB pairs were uttered by different speakers (see Fig. 3.2), and only 18% deemed SBB pairs to be from the different speakers, indicating the high quality of our responses. For SAB pairs, only 50% of participants deemed the speakers different. Compared to SBS pairs, we still see a degradation as the detection probability for those is 30%, which is expected since our transformations manipulate the spectra of the samples. However, even after the degradation, detection is only as good as a random guess (50%), demonstrating the inability to spot our adversarials. In comparison, FakeBob reports results for two types of adversarials; effective and ineffective. Effective adversarials remain successful after being played over the air, while ineffective ones lose this ability. For effective adversarials, FakeBob reports that 54% of their participants could tell that they belonged to a different speaker (compared to the benign sample in the pair),

while for ineffective ones, this rate drops to ∼30%.

Our adversarials (50% detection) outperform FakeBob's effective adversarials (54% detection), highlighting their quality. When comparing our results (50% detection) to FakeBob's ineffective samples (30% detection), it is important to note that our study is more restrictive; participants were informed that some samples are machine-generated, making them more vigilant and harder to fool [261], as opposed to FakeBob, wherein the study's purpose is never disclosed to participants. Second, FakeBob's study contained more samples per category, whereas ours has more categories but fewer samples in each, making a single "bad" frequently-spotted adversarial more influential. Considering this and our adversarials' 50% detection avoidance rate, the study proves our attack's strength.

3.9 Discussion

Practicality. Our attack takes around 4 seconds per attempt and requires a limited number of attempts, making it highly practical. The attacker first needs to generate spoofed samples, which is instant with modern VC/SS algorithms [184].

Potential defenses. *Traditional adversarial defenses:* These include methods such as spatial smoothing, audio squeezing, and adversarial training [266]. FakeBob [37] shows smoothing and squeezing are ineffective against attacks on acoustic systems, and adversarial training and similar methods [193, 121, 252, 52, 133] lose effectiveness against unseen attacks [151]. We also believe our attack is naturally robust to many such traditional defenses against optimization-based adversarials since it conceptually deviates from the hypotheses behind them.

Liveness detection: As explained in §3.2.2, *active* liveness detection methods are hypersensitive and therefore less practical. Other non-traditional approaches include Voicefox [225], which suggests that higher mistranscription rates by STTs may indicate that a sample is spoofed but the quality of modern spoofing algorithms and the pace at which they develop will soon make such methods invalid. Blue et al. [25] detect spoofed speech through vocal tract reconstruction. However, vocal tract manipulation tools [48] may defeat this defense.

Attack-specific defenses: We considered defenses specific to our attack. The first is eliminating silence (applying VAD), nullifying the effects of "non-speech" cues that we inject in F_1&F_2. Although we state that these intervals are crucial for CMs (see §3.4.2), Zhang et al. [296] propose a model ($SENet_FFT$) capable of classifying silence-free speech. We implement it but find applying VAD leads to poor performance on the test set even without

our attack (EER=24.34%). We attribute this to us using a well-known VAD algorithm ($WebRTCVad$ [263]), which is aggressive at filtering out silence, while they provide no details regarding their probably much more permissive VAD algorithm, which leaves many silence intervals in the samples, enabling proper functionality. Thus, VAD is not a viable defense. Noise reduction will similarly filter out vital silence information.

The second defense is low-pass filtering. We boost the center of the spectrum, but higher components are amplified as well. These should not be considered for audio classification as they are noise-dominated, unreliable and missing in bandwidth-limited scenarios (see §3.7). Yet, this byproduct could be a key reason behind the attack's success, which can be defeated by rejecting information in this range. Zhang et al. [296] also suggest using $SENet_FFT$ with frequencies above $4KHz$ discarded to increase robustness. We reproduce this, and achieve a comparable EER—2.11% (without VAD). Afterward, we run our attack. For spoofed samples, the model is robust, with a success rate of 0.63%. Yet, with our attack, the rate rises to 11.18% (single attempt), proving our spectral manipulation effects (F_3&F_5) are deeply rooted in the central regions of the spectrum, where relevant information resides.

Identifying promising VA improvements: The best evaluated defense against our attack comes in the form of a CM already evaluated (WAV2VEC), for it having the potential to reject background noise and high frequency components. We explain this in §3.6.1, but demonstrate how even this defense falls short. The over-telephony setting, however, reveals additional interesting findings: transcoding together with the bandwidth loss, combined with training on noise-augmented data (e.g., WAV2VEC) can lead to a degradation in the attack's success (see §3.7). We believe that this is a promising research avenue to explore in the search for robust spoofing countermeasures, both in the designated-app setting and for phone-call authentication. Despite the fact that the loss incurred by transcoding and down-sampling is a byproduct of transmission over the telephony network, these components can be simulated algorithmically, potentially raising the bar for attackers who are currently more easily capable of mounting attacks in the designated-app scenario, especially when combined with channel invariant models (obtained by training on noise-augmented data) such as WAV2VEC. However, at this embryonic stage, our attack still achieves concerning success rates after a number of attempts against this configuration (see §3.7).

Outlook. Spoofing detection is a cat-and-mouse game, wherein novel algorithms and threats are emerging at an unprecedented rate without a universally reliable solution. Even the threat of replay attacks alone is still unaccounted for (in text-dependent schemes) despite many works describing potential solutions in top venues [6, 254], as new studies are constantly demonstrating how slightly different settings or audio channels [197, 261] and attacks [254] can severely degrade the performance of such systems. The facts are similar for fake speech detection, as despite the tremendous efforts invested so far, reports discussing

the lack of generalization to different settings and algorithms continue to appear [7, 296, 161]. Nonetheless, replay/spoofing detection systems haven been gaining popularity in practice due to the lack of better alternatives. While recent works present somewhat generalizable models (e.g., WAV2VEC), their true ability to extend to all settings remains an open question. Regardless, we attacked the state-of-the-art available *passive* spoofing detectors, which are suitable for and being used in security-critical deployments, and proved their vulnerability to our transformations under the assumptions where they perform best. In the future, it is imperative to study the link between our findings and generalization.

3.10 Related Work

Non-Proactive Spoofing Attacks. These attacks fall under four sub-classes: replay [268], mimicking [145], voice conversion (VC) [134], and speech synthesis (SS) [223] (see §3.2). Replay and mimicking are often disqualified as ASVs can be easily made robust against them (see §3.2). We restrict the discussion to SS/VC attacks, which are successful in fooling ASVs, achieving success rates up to 100% [261]. This motivated the development of (CMs) [269, 6] to protect ASVs. Many advanced CMs have since appeared, demonstrating robustness and generalization [269, 241, 277, 6, 235], and such CMs are deployed by leading VA vendors [115]. Thus, we disqualify spoofing attacks assuming CMs defeat them.

Adversarial Attacks on ASVs and STTs. In targeted adversarial attacks on ASVs, the attacker starts from some audio sample (typically, but not necessarily, a human recording), and adversarially optimizes it to fool the ASV into believing it was uttered by the victim. Against STTs, the task is to make them transcribe the sample into text of the attacker's choosing.

The literature describes many such attacks. However, they fail to break VA under security-critical deployment constraints. First, many assume (semi-)whitebox access (at least some crucial parameters are known) to the target models—STTs [33, 45, 283] , ASVs [274, 156, 253, 138, 152, 247, 117], or both [157]. They all either fail to attack blackbox systems or rely on transferability to do so. Yet, adversarial transferability across ASVs/STTs has been proven very limited [3], and works demonstrating good transferability rates (for targeted attacks) only do so when their targets and shadows share many commonalities, making them unsuitable in practice. Other approaches attacking STTs [10, 42, 300] and ASVs [37, 166, 300] circumvent the lack of transferability via target-specific attacks that repeatedly query the model to infer its decision boundary. These are query-inefficient and do not fit our threat model. Recently, NI-OCAAM [300] satisfied the small query budget requirement. Nonetheless, the technique is based on CommanderSong [283], which

is extremely slow, violating the response time constraint (see §3.2). The same issue is faced by SirenAttack [72].

Another class of attacks operates at the signal processing level, such as Mystique [7], which fools ASVs via acoustic resonance perturbations. While Ahmed et al. [7] conduct an experiment demonstrating that Mystique could even bypass detection by CMs as well if they are not adversarially trained on samples similar to the attack inputs it generates, Mystique requires access to the probability scores of the target ASV and external devices, and is not query-efficient. Abdullah et al. [1] target ASVs/STTs and show high success rates. Yet, the attack is untargeted and can only make a model arbitrarily misclassify, which is of no use in our case. Abdullah et al. [2] show how certain transformations can make audio impossible to understand by humans, while maintaining correct classification by ASVs (STTs). However, they consider a different goal, which is hiding a *previously-recorded* bonafide sample of the victim in unintelligible audio to deceive humans, while we do not assume access to such recordings, as the phrases to repeat are generated randomly by the target (see §3.2).

Adversarial Attacks on CMs. Adversarial attacks on CMs are advanced attacks that combine spoofing with adversarial examples and are conceptually similar to our own. They add adversarial perturbations to SS/VC outputs to get CMs to misclassify them as bonafide. Due to the vulnerability of ASVs to spoofing and the high quality of generative algorithms, such perturbations, when subtle enough, can make spoofed speech bypass all VA components combined. However, similar to adversarial attacks on ASVs/STTs, known attacks on CMs still have not achieved this goal under security-critical deployment constraints.

Most attacks on CMs (or joint ASV-CM systems) [160, 295, 293, 292] have been whitebox and thus of limited applicability, or rely on transferability to attack blackbox systems. However, transferability attacks against CMs, as is the case with ASVs/STTs, are extremely limited in potential and we attribute the observed high transferability rates to not considering fully blackbox settings and experimenting with small sets of CMs lacking robustness in real-world settings (see §3.6.1). Finally, Ding et al. [68, 67] propose a novel method for adversarially enhancing VC outputs to bypass CMs. Yet, their attack is only (semi-)whitebox and they experiment with a single CM. There are target-specific attacks that require many target queries [119, 85] to estimate the decision boundaries. These attacks are ineffective under security-critical deployment constraints (see §3.2). In a concurrent work, Hua et al. [112] propose enhancing VC outputs to spoof CMs by adding global noise and replacing the silent intervals in the generated speech with ones from bonafide samples in a post-generation phase. We integrate similar (yet different) stages into our transformations (see §3.4.2). However, the improvement of these alone is not drastic, and other steps are vital to generate robust adversarials.

3.11 Conclusion

We presented the first practical attack on security-critical VA deployments, by targeting a newly-identified weakest link— CMs. Our novel targeted, real-time, and model-agnostic attack generates high-quality fake speech, fooling both machines and humans, and enabling the attacker to impersonate the victim, severely compromising users' security. Our attack's success is concerning primarily due to its blackbox nature, and practicality—including the designated-app and phone conversation scenarios. Our findings highlight the pitfalls of modern VA systems and the need for defenses.

Chapter 4

UnMarker: A Universal Attack on Defensive Image Watermarking

This chapter is adapted from work published at the IEEE Symposium on Security and Privacy (S&P) 2025 [130]. Building on Chapter 3, we revisit the same deployment assumption—that relying primarily on heavily-restricted access to model architecture and internals can provide a strong basis for security—but now in the context of image watermarking, a proactive deepfake detection paradigm that fundamentally relies on the secrecy of the embedded signal and its detection mechanism. We show that this assumption fails by analyzing how watermarking schemes encode information across images and identifying a universal spectral carrier intrinsic to this process. This reframes watermark removal as a signal disruption problem over invariants shared by all watermarked images. Leveraging this insight, we develop *UnMarker*, a *universal, black-box, query-free, and data-free* attack that removes watermarks through quality-preserving adversarial optimization. *UnMarker* breaks seven state-of-the-art schemes—including *semantic* watermarks previously considered robust and viewed as promising candidates for future deepfake detection—reducing the best detection rate to 43%. Beyond academic impact, it was later shown to break *Google*'s *SynthID* [91], recognized through Google's Vulnerability Reward Program, and has since been widely adopted as a benchmark for evaluating watermark robustness [149, 220, 137, 116, 199, 90].

4.1 Motivation

Generative AI (*GenAI*) has amassed popularity due to its growing sophistication and accessibility [88, 46, 190]. This technology has solved problems in various fields [75, 231, 55,

144]. Yet, it comes with the risk of malicious entities using it to generate harmful images (deepfakes) [61, 104], proven by the many reports of those being used in political smear campaigns [5] and non-consensual pornography [99].

Defenses first emerged as passive detectors [118, 150, 81, 297] that tell apart fake from real images, but these are often ineffective, failing to generalize to unseen synthesis algorithms [49]. As training advanced generators is costly and prohibitive [128] in the absence of the extensive resources and high-quality data that only a few *GenAI* providers possess, adversaries typically resort to these providers' *API*s to create harmful content. This recently led to the adoption of defensive watermarking as a countermeasure, which enables providers to embed hidden signatures in their images that can later be identified using designated detectors. Defensive watermarking has gained traction in the research community [260, 303, 78, 282, 281, 164, 236, 299] and the industry, as providers have already implemented and are testing such solutions (e.g., *Google*'s *SynthID* [91]), or pledged to do so soon [23] in commitments to the *White House* [101].

Defensive watermarking protects against limited adversaries, following the assumption that attackers lack the resources to train complex generators. Yet, existing schemes do not currently demonstrate strong guarantees for this resistance beyond empirical robustness to naive manipulations. We devise *UnMarker*— the first *universal* (effective against **all** schemes), *black-box* (without access to the scheme's parameters or similar systems), *data-free* (no additional data needed), and *query-free* (without detector feedback) attack.

Our motivating insight is that, for any scheme, there must be a ***universal carrier*** that embeds the watermark in all images. In layman's terms, this *carrier* is the set of all attributes the detector may examine when verifying the watermark. This set must exist and is finite (shared among all images) as detectors are computationally limited. By analyzing the desired resistance to basic manipulations and the requirement that watermarks must naturally integrate with various contents, we deduce that this *carrier* can ***only*** be in the global spectral amplitudes. That is, a "conventionally" robust scheme *must* construct its watermark by manipulating the frequencies at which the different pixels vary across the entire image, forcing the *collective* distribution of these value shifts at different rates to follow a unique pattern. We can roughly even isolate the range in which watermarks are encoded: the more robust *semantic* watermarks [260, 236], which alter the structure, target low frequencies, while content-preserving *non-semantic* watermarks [282, 303, 164, 78] populate higher bands.

UnMarker disrupts these spectral amplitudes. To remove *non-semantic* watermarks, it *directly* modifies the image to optimize a designated loss, affecting the rates at which *individual* pixels vary w.r.t. their neighbors and impacting the distribution of high-frequency

amplitudes. For *semantic* watermarks, we devise novel optimizable filters and a procedure to learn their weights to *systemically* alter the consistency at different regions, causing discrepancies at lower frequencies. Unlike traditional filters that rely on geometric heuristics, limiting them to minor distortions, *UnMarker* is guided by a perceptual loss to maximize the disturbance in visually non-critical areas without compromising quality. As a result, the *carrier* set of frequency bands encoding the watermark is shifted away from the watermark's distribution. By targeting the *carrier*, *UnMarker* abandons the dependence on a specific model, leaving it successful without knowledge of the scheme and without detector feedback. Previous adversarial attacks [214, 120, 15, 165, 110, 249] all face this limitation, failing under realistic conditions.

We experiment with seven *SOTA* schemes from top venues. By combining the above techniques with *mild cropping*, *UnMarker* breaks them all, despite its *black-box* and *query-free* nature. We compare *UnMarker* with the recent *VAEAttack* [298] and *Diffusion-Attack* [298, 214, 15], demonstrating far superior results, defeating *semantic* watermarks against which they fail and disputing misconceptions regarding these schemes' robustness to limited adversaries [298, 15, 214]. Our results urge vendors to rethink their response to deepfakes. We make the following contributions:

- We thoroughly analyze defensive watermarking and reveal trade-offs that make it an invalid countermeasure.
- We employ novel spectral optimizations to devise *UnMarker*— the first *universal*, *data-free, black-box*, and *query-free* attack against defensive watermarking[1].
- We demonstrate *UnMarker*'s efficacy against seven *SOTA* schemes under restrictive settings, rendering them useless.
- We compare *UnMarker* to the strongest *query-free black-box* attacks, attaining superior performance and quality.
- Using a novel class of optimizable filters to incur structural disturbances, *UnMarker* is the first practical attack to defeat *semantic* watermarks, whose robustness has marked them as the ultimate candidates for trustworthy deepfake detection, showing this assumption is highly questionable.

[1] https://github.com/andrekassis/ai-watermark

4.2 Background & Related Work

We refer the reader to §2.2 for a general overview of content authenticity mechanisms and defensive watermarking. At a high level, a watermarking system consists of an encoder $\mathcal{E}_\theta$ that embeds a watermark into generated content and a detector $\mathcal{D}_\phi$ that verifies its presence. Existing image watermarking schemes can be broadly categorized as *general-purpose* (post-hoc) or *generator-specific*, depending on whether watermarking is applied to existing images or integrated into the generation process. In this section, we further classify these approaches into *semantic* and *non-semantic* schemes based on how the watermark influences the generated content. We then review prior attacks on defensive watermarking and highlight their limitations, which motivate the removal strategies developed in this chapter.

4.2.1 *Semantic* vs. *Non-Semantic* Watermarking

***Non-Semantic* Schemes.** These methods embed watermarks in a manner that does not meaningfully affect the visual content of the image. Formally, given a fixed conditioning input $\boldsymbol{c}$, the outputs $\mathcal{E}_\theta(\boldsymbol{c}, \boldsymbol{w}_1)$ and $\mathcal{E}_\theta(\boldsymbol{c}, \boldsymbol{w}_2)$ corresponding to different watermarks $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ remain visually (i.e., perceptually) indistinguishable. This property typically also holds for *general-purpose* schemes where the watermark is fixed and implicitly encoded in the model parameters—see §2.2.2, i.e., $\boldsymbol{x}_w = \mathcal{E}_\theta(\boldsymbol{c})$ remains visually (almost) identical to the original, non-watermarked input image ($\boldsymbol{x} \equiv \boldsymbol{c}$). Aside from *StegaStamp* [236], *general-purpose* schemes are *non-semantic*. Many *generator-specific* schemes, including those considered in this work [282, 281, 164, 78], are also *non-semantic*, as their training explicitly enforces invariance of the generated content with respect to the watermark.

***Semantic* Schemes.** *Semantic* watermarking, in contrast, allows the watermark to directly influence the generated content. This term was popularized by Wen et al. with their introduction of *TRW* [260], which alters the latent noise in text-to-image *Latent Diffusion Models (LDMs)* [211]. When provided with the same textual prompt but different random latent codes (noise), the *LDM* generates images that satisfy the prompt while differing in content. By injecting structured patterns into these latent codes, *TRW* produces watermarked images that remain consistent with their non-watermarked counterparts (i.e., those obtained from the same latent noise without modification) due to the shared conditioning prompt and latent initialization, while exhibiting visible differences in the rendered content induced by the modified codes. This preserves the naturalness of the

resulting watermarked content and avoids undesired visible artifacts while leaving the watermarks highly robust to existing removal attacks [298, 214, 15].

Although other generator-specific watermarking schemes following similar design principles have emerged recently [132, 135, 279, 95], TRW was the primary representative at the time of the original work and is therefore used as the canonical example in this chapter.

StegaStamp [236] precedes *TRW* and operates similarly to its *general-purpose* counterparts. Yet, compared to typical *general-purpose* schemes, it is less stealthy, causing structural changes (see Fig 4.2). Still, its strategically distributed distortion helps these changes blend somewhat naturally into the image. In defensive watermarking, users cannot observe non-watermarked images, further hiding these nuances and leaving *StegaStamp* applicable. While *semantic watermarks* primarily refer to *TRW*, previous work recognizes the similarities between both schemes, distinguishing them as more robust [298, 214, 15]. Thus, we refer to *TRW* and *StegaStamp* as *semantic* watermarking schemes in this work.

4.2.2 Attacks Against Defensive Watermarking

Regeneration Attacks

Proposed by Zhao et al. [298], they rely on *Diffusion Models (DMs) [227]* or *Variational Autoencoders (VAEs) [133]* (henceforth, the *DiffusionAttack* and *VAEAttack*) to introduce noise and then reconstruct the images after the watermark has been eliminated. Saberi et al. [214] and *WAVES* [15] revisit these attacks. Regeneration is the only practical (*black-box* and *query-free*) attack against defensive watermarking. All these works verify the vulnerability of *non-semantic* watermarks, against which regeneration offers theoretical removal guarantees. However, regeneration fails against *semantic* watermarks. For instance, while *TRW* experiences some degradation [15], it occurs at a considerable distortion where the attack becomes invalid (see §4.3). Regeneration also relies on the denoising pipelines' fidelity, often impacting quality (see §4.6.2). *UnMarker* defeats all watermarks while also abandoning the need for pretrained models, ensuring quality.

Adversarial Attacks

Realistically, attackers do not have white-box access to the scheme (see §4.3), making many traditional adversarial attacks infeasible. Thus, existing works make relaxing assumptions, failing to retain success under practical conditions.

WEVADE [120] attacks watermarked images by querying the detector and updating them accordingly until the watermark is not detected. Yet, detector feedback may not be available (see §4.3). Lukas et al. [165] propose *query-free* attacks but assume unrealistic access to a generator with similar architecture and data (see §4.3). Saberi et al. [214] mount an attack on a surrogate detector, which transfers to the target. However, training the substitute requires non-watermarked samples from the provider's distribution, violating the defensive watermarking principles [15, 110]—see §4.3—as providers only release *watermarked* images and possess exclusive datasets. Additionally, it still fails against the *semantic StegaStamp* without affecting quality. *RD-IWAN* [249] also requires access to non-watermarked provider images. An et al. [15] evaluate surrogate attacks specifically on *semantic* schemes, finding *StegaStamp* robust, while *TRW* is only affected under unrealistic access to its autoencoder or images generated by it with *different* watermarks, which is impractical as any provider's published image bears its *single* identifying watermark. They also prove Saberi et al.'s [214] surrogate attack unsuccessful absent non-watermarked provider images. Hu et al.'s [110] ensemble surrogate attack requires no such non-watermarked images. Yet, their evaluations exclude *semantic* watermarks and only consider surrogates and targets of the same scheme, leading to sufficient commonalities for transferability and violating the *black-box* setting. It also requires extensive resources (40–100 surrogates), data, and large perturbations, making it impractical. *UnMarker* waives all impractical assumptions, requires no data or prohibitive resources, and preserves quality.

4.3 Threat Model

Our framework considers two parties: ***Providers*** who develop and monetize *GenAI* platforms, and ***Attackers*** abusing their access to generate malicious content.

Providers' Goals. *GenAI* providers must ensure their images are not used to spread disinformation. Thus, they watermark generated images. For a suspicious image, they invoke the detector to test for the watermark, proving it fake.

Providers' Capabilities. They have resources and data to train *SOTA* generators. As is common practice [88, 190], their *API*s are restrictive, accepting queries (e.g., text) and returning watermarked images, while hiding system internals and intermediate results.

Attacker's Goals. Attackers wish to generate real-looking deepfakes by invoking the providers' *API*s and then using methods to 1) remove watermarks while 2) retaining quality.

Attacker's Capabilities: *Limited Resources.* Following defensive watermarking assumptions (see §2.2.2), attackers lack high-quality data or extensive resources. Thus, training

advanced models or surrogates for the providers' generators [165] or detectors [214] is not possible. As some services (e.g., *AWS*) allow renting advanced hardware, attackers may access powerful machines. Yet, this access can only be short-term due to high costs, not for training large models over long *GPU* hours.

Black-Box. As many providers keep their algorithms and datasets secret [215], attackers have no white-box access to (similar) models, data, or any information about the system.

Query-Free. When providers restrict access to their detector to trusted corporations or employees, as confirmed by recent announcements [60], attacks relying on detector feedback [120] lose their efficacy. Given these announcements and defensive watermarking's vulnerability when query access to the detector is publicly available (see §4.2.2), we follow previous work [110, 298, 214] and investigate this paradigm against *query-free* attacks where attackers cannot interact with the detector directly to observe its decisions.

4.4 Design Philosophy

Since the detector $\mathcal{D}_\phi$ must be able to verify the watermark in ***any*** image, it should know where to find this watermark, meaning there must be a set of attributes, namely the *universal carrier*, that can be scanned in all images to make this decision. Our analysis will reveal that identifying this set is possible since the requirements from a robust scheme eliminate all options except ***the global spectral amplitudes***. Thus, any robust scheme will inject its watermarks in the collective magnitudes of the different frequency bands. In §4.5, we show how identifying this *carrier* enables attackers to mount destructive attacks that remove the watermark with minimal visual impact and no knowledge of the system.

4.4.1 Uncovering The Universal Carrier

Any computationally limited detector $\mathcal{D}_\phi$ can only monitor a finite set of attributes $\mathcal{P} = \{p_b |\ b \in [\![B]\!]\}$ in all images. Deciding if an image is watermarked relies on whether (a subset of) $\mathcal{P}$'s attributes follow a *watermark property distribution* $\mathcal{P}_w$ that $\mathcal{D}_\phi$ associates with the watermark w. While $\mathcal{P}$ is a set of attributes, $\mathcal{P}_w$ is a distribution over $\mathcal{P}$'s values, that identifies w. Note that there may be several patterns $\mathcal{P}_w$ pertaining to subsets of $\mathcal{P}$. We term $\mathcal{P}$ the *universal carrier* as it contains all the properties w can influence. Although $\mathcal{P}_w$ should allow watermarking different contents, its variance must still be sufficiently low to prevent detecting w in non-watermarked images, restricting it to a reasonably narrow vicinity. Thus, we conjecture that introducing even relatively small deviations into $\mathcal{P}$'s values can cause

any watermarked image to violate this distribution $\mathcal{P}_w$, removing the watermark without degrading quality. This destructive strategy requires no knowledge of $\mathcal{P}_w$ itself, making it feasible without feedback from $\mathcal{D}_\phi$ if the *carrier* $\mathcal{P}$ is correctly identified. To identify $\mathcal{P}$, we start by inspecting widely-accepted requirements [260, 164, 303, 236, 282, 281, 78] from a robust scheme:

Robust Watermarking Requirements. *Universal Applicability:* The scheme must be capable of watermarking images of various concepts and scenes, and the watermark must always be detectable.

Universal Stealthiness. The process cannot introduce unnatural artifacts that degrade quality. We do not require the watermark not to leave visible traces since *semantic* watermarks violate this concept by definition, influencing the structure. Yet, the quality of the watermarked outputs remains crucial for the utility and usability of the scheme.

Robustness to Manipulations. The watermark should resist—at least—naive manipulations (e.g, *cropping*, *blurring*, *compression*).

Candidates

An image is characterized by its 1) *spatial structure* (pixel values), and 2) *frequency distribution*— the rates at which values alternate between neighboring pixels. Thus, the *carrier* must reside in a channel offered by these media. Below, we present hypotheses, accompanied by an in-depth analysis, regarding the possibility of injecting watermarks in each of these channels, identifying a unique *carrier* capable of satisfying all demands. Our claims are corroborated by *UnMarker*'s empirical success in §4.6.

H1. ***Watermarks cannot be embedded in the spatial structure (pixel values)***. This is because they make for an *unreliable carrier* that experiences sporadic variations (in the absence of a watermark) between images based on their contents (different images assign extremely different values to their pixels). However, the scheme must always bring the *carrier*'s values to the same distribution $\mathcal{P}_w$ for any watermarked image. When these values naturally (i.e., without the watermark) exhibit a high degree of content dependency, this is impossible without restricting the scheme to processing a small set of similar contents or leaving unnatural cues in many outputs corresponding to contents that would otherwise require these values in images portraying them to lie far away from the desired $\mathcal{P}_w$. As this distance increases, the magnitudes of the changes needed to bring them to $\mathcal{P}_w$ grow, failing to integrate naturally with the content. Content restrictions violate *applicability*, while unnatural artifacts are not *stealthy*. A scheme may attempt to strategically place pixels encoding the watermark to accommodate large changes with minimal visibility. Two

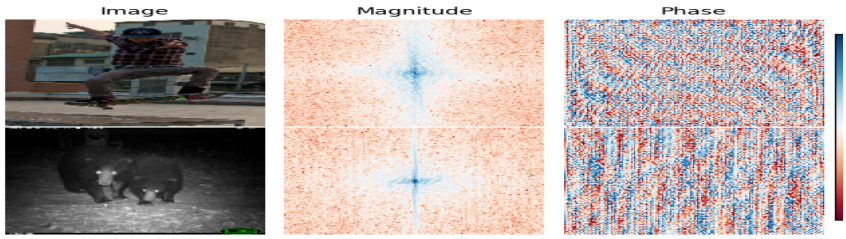


Figure 4.1: Spectral analysis of two images. Both depict worldly objects of high consistency. Thus, the ***collective*** spectral magnitudes are distributed similarly as the low frequencies (center) corresponding to gradual variations are always far more dominant. Phases determine the spatial arrangement of the pixel value shifts that contribute to these magnitudes to shape the content, making them extremely different.

possibilities exist: either they are positioned far apart to remain unnoticeable despite significantly differing from their environments, or located in regions the main content is unlikely to occupy (i.e., the background). Both are volatile and cannot withstand manipulations (e.g., *cropping* or *noise*), violating the *robustness* requirement (see §4.6.2).

We now turn our focus to the frequency distribution. There are two options for influencing frequencies to inject watermarks: Locally or globally. We claim the following:

H2. ***Local changes to the pixel shift rates cannot reliably encode watermarks***. The reason is that local modifications affect the rates at which pixels vary in select spatial regions. These share the above pitfall of strong dependency on the content, leaving only global spectral changes.

The spectrum associates frequencies with two properties: *magnitude* and *phase*. Magnitudes represent the prevalence of shifts that occur at the specific rate across the image. Mathematically, any image is a superposition of periodic signals with varying frequencies occurring at different phases that determine their orientations, forming the image's structure based on how they intersect. Accordingly, and similar to the spatial structure, we deduce that:

H3. ***Phases are an unreliable carrier due to their high sensitivity to the content***. This is the result of the phases varying drastically between images as the primary factor affecting the structure (see Fig. 4.1).

By process of elimination, hypotheses **H1-H3** exclude all other candidates, leaving only one potential *carrier*: The global spectral magnitudes. These are collective quantities with little dependency on specific pixels, making them resistant to reasonable manipulations that

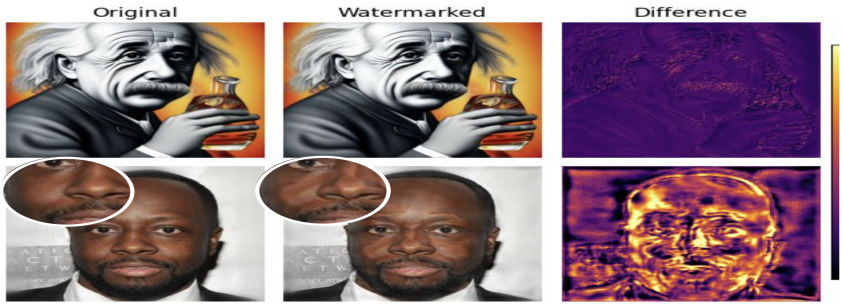


Figure 4.2: Images watermarked by *StableSignature*— *non-semantic* (top), and *StegaStamp*— *semantic* (bottom). The rightmost figures display the differences between the original and watermarked images that correspond to the changes encoding the watermarks. *StableSignature*'s modifications are restricted to existing (high-frequency) edges such as wrinkles, hair, mustache, and intersections of multiple components. *StegaStamp*'s watermark is distributed across the image. The magnified area shows how it manipulates the consistency (texture), injecting gradual (low-frequency) changes that manifest as wrinkles at this location.

would preserve the majority of energy (amplitude) distribution. They are also distributed similarly for any realistic content: Lower frequencies contain the majority of energy as they correspond to gradual (slow) changes that capture the inherent consistency (continuity) of the worldly objects images depict. Higher frequencies (rapid transitions) occur at edges (intersections of objects), making them far less prevalent (see Fig. 4.1). Thus, while amplitudes vary between images, they always follow a similar distribution, decaying as the frequencies grow. This stability other alternatives lack allows injecting watermarks via changes of similar magnitudes for any content, enabling them to be small s.t. they always integrate naturally.

Spectral Watermarks

The above entails that (conventionally) robust watermarks must reside in the global spectral amplitudes. Based on the different schemes' characteristics, we now present our main hypothesis, which serves as *UnMarker*'s foundation:

H4. ***Non-semantic schemes primarily inject watermarks in the global spectral amplitudes associated with high frequencies, while semantic watermarks mainly target low frequencies***. Below, we analyze this claim:

Non-Semantic: As these watermarks cannot influence the content, they are typically injected in high frequencies since sufficiently manipulating lower bands s.t. the watermark is

detected requires substantial changes due to the initial vast energy concentration in this range. To alter the collective distribution at high frequencies, polarization at ***existing*** edges can be controlled by adjusting the intensities of the various intersecting components to amplify or attenuate the transitions between them. This significantly impacts higher frequency amplitudes via small (invisible) modifications only due to the naturally low energy distribution in this range and by exploiting the characteristics of these regions (edges) that already exhibit high-frequency shifts to mask the watermark's footprint. Fig. 4.2 (top) depicts this behavior for the *non-semantic* *StableSignature* [78]. We verified that other *non-semantic* schemes operate similarly.

Semantic: Since these schemes can influence the content, they are capable of populating the consistency-defining low frequencies, entangling their watermarks with the structure of the image and making them harder to eliminate. Unlike *non-semantic* watermarks, they are not restricted to edges, affecting various objects' structures and textures. As spectral amplitudes represent the *collective* contributions from pixel variations occurring at the relevant rates from all spatial locations, these schemes can construct their outputs s.t. the needed changes are spread among different regions, reducing undesired visual artifacts. Yet, since the changes required to alter low-frequency magnitudes are large, their influence may be visible. Fig. 4.2 (bottom) portrays this for *StegaStamp* [236]. *TRW*'s [260] images may display different content from the original (non-watermarked), making illustrations uninformative (see §4.2.1). Yet, our results in §4.6.2 prove its watermarks are *primarily* in low frequencies.

Remark (On the *semantic* vs. *non-semantic* distinction). In §4.2.1, we classified watermarking schemes as *semantic* or *non-semantic* following prior work: a scheme is *semantic* if the **watermark** influences the generated content, and *non-semantic* if watermarked outputs remain perceptually indistinguishable from their non-watermarked counterparts. While this taxonomy was sufficient for existing methods at the time of writing, it may not fully capture emerging designs, as the field is evolving rapidly [132, 135, 279, 95]. In particular, the studied content-influencing schemes [236, 260] exhibit *semantic* behavior and predominantly embed their watermark signal in lower-frequency components, as we show in §4.6.2, which contributes to their robustness against prior removal attacks [298, 214, 15], as explained above. However, there is no fundamental requirement that future content-influencing schemes must follow this pattern. A scheme may alter the content while still embedding its detectable watermark signal primarily in higher-frequency components. Such schemes do not align cleanly with the traditional *semantic*/*non-semantic* distinction: they are not *semantic* since their measurable watermark footprint itself is in higher frequencies and thus is not what governs the content, yet are not *non-semantic* due to observable

output differences. More generally, watermark signals need not concentrate in a single frequency regime. For instance, as we show empirically for *TRW* in §4.6.2, despite being a *semantic* scheme with most of its watermark footprint residing in lower frequencies, a portion of it still populates higher bands. Future schemes may distribute this footprint more evenly across the spectrum.

To resolve this ambiguity, we emphasize that *UnMarker*'s attack stages are driven by the spectral footprint of the watermark rather than its effect on content. While the stages in §4.5.1 and §4.5.2 are framed as targeting *non-semantic* and *semantic* schemes, they in effect operate over watermark signals concentrated in high- and low-frequency bands, respectively. In particular, all schemes whose watermark footprint lies predominantly in higher bands are effectively handled by the high-frequency stage. *Even if a (future) scheme exhibits a substantial watermark footprint across both low- and high-frequency components, without a clear dominant concentration, the combined application of these stages aims to effectively remove watermark signals across the spectrum.*

4.5 UnMarker

Now that we have identified the carrier, we present our universal attack. Our objective is to modify watermarked images so that no detector can identify the watermark, while retaining quality and similarity. However, access to detectors or their gradients is not available, making direct instantiations of the adversarial optimization problem formulated in **Definition** 2.3.1 impossible. As such, similar to the broader adversarial attack framework discussed in §2.3.1, we instead optimize a spectral surrogate objective informed by the analysis presented thus far in this chapter to achieve this outcome under the threat model's restrictions.

It follows from §4.4.1 (**H4**) that the *carrier* must be a set $\mathcal{F}_c = \{|F_b| : b \in [\![B]\!]\}$ of spectral amplitudes, and the *watermark property distribution*, denoted as $\mathcal{F}_w$, is over these amplitudes. Let ℓ_d be a perceptual distance metric s.t. if $\ell_d(\boldsymbol{x}, \boldsymbol{v}) \leqslant t_{\ell_d}$ for some threshold t_{ℓ_d}, then $\boldsymbol{x}$ and $\boldsymbol{v}$ are visually similar. We can now formalize watermark removal:

Watermark Removal. *(WR): Given watermarking algorithm $\mathcal{A}$ whose carrier frequency set is $\mathcal{F}_c = \{|F_b| : b \in [\![B]\!]\}$ with watermark property distribution $\mathcal{F}_w$, the watermark $\boldsymbol{w}$ is removable from $\boldsymbol{x}_w$ if there is an image $\boldsymbol{x}_{nw}$ s.t. $\ell_d(\boldsymbol{x}_w, \boldsymbol{x}_{nw}) \leqslant t_{\ell_d} \wedge \mathcal{F}_c(\boldsymbol{x}_{nw}) \nsim \mathcal{F}_w$.*

Removing the watermark from $\boldsymbol{x}_w$ amounts to finding a visually similar $\boldsymbol{x}_{nw}$ for which $\mathcal{F}_c$'s attributes deviate from the expected distribution $\mathcal{F}_w$. Due to our black-box framework, the specific $\mathcal{F}_c$ is unknown. Instead, we seek $\boldsymbol{x}_{nw}$ that retains visual similarity but maximizes

some generalized spectral difference ℓ_F from $\boldsymbol{x}_w$:

$$\boldsymbol{x}_{nw} = \underset{\boldsymbol{x}'}{argmax}\ [\ell_F(\boldsymbol{x}_w,\ \boldsymbol{x}')] \quad s.t.\ \ell_d(\boldsymbol{x}_w,\ \boldsymbol{x}') \leqslant t_{\ell_d} \tag{4.1}$$

Maximizing this difference across **a broad but sufficiently restricted set of frequencies** that include $\mathcal{F}_c$ should lead to significant discrepancies in the values of $\mathcal{F}_c$ as well, forcing them to deviate from $\mathcal{F}_w$. As *semantic* and *non-semantic* watermarks target different parts of the spectrum, we must craft a suitable attack strategy for each, identifying a proper loss function and optimization procedure.

4.5.1 Attacking Non-Semantic Watermarks

We now instantiate *eq.* (4.1) to defeat *non-semantic* watermarks, while §4.5.2 presents a method against *semantic* schemes. Following §4.4.1, *non-semantic* watermarks are in higher frequencies. Thus, we must disrupt this range without leaving visible cues. We present our spectral loss *DFL* that is suitable for high frequencies and the visual loss ℓ_d, explaining how each optimization step updates the input to finally arrive at a watermark-free output.

Spectral Loss. We propose the *Direct Fourier Loss (DFL)*:

$$\forall\ \boldsymbol{x}, \boldsymbol{v}: \qquad DFL(\boldsymbol{x},\ \boldsymbol{v}) = \|FT(\boldsymbol{x}) - FT(\boldsymbol{v})\|_1$$

where FT is the $(2D)$ Fourier transform. *DFL* calculates the band-wise spectral differences between $\boldsymbol{x}$ and $\boldsymbol{v}$, outputting the sum of their magnitudes. Optimizing *DFL* maximizes the amplitudes of the differences in the collective spectral representations between the sample being optimized and the watermarked image $\boldsymbol{x}_w$, which leads to an output whose spectral magnitudes differ from $\boldsymbol{x}_w$'s. Since the watermark is encoded in these amplitudes, it will be disrupted, causing the detector to fail to recognize it. Yet, this is only guaranteed if the perturbation budget that controls the quality and similarity to the watermarked image is permissive enough for such significant distortions to occur. Luckily, this is true even under such small budgets when disturbing high frequencies since the energy concentration in these bands—and accordingly, the watermark's footprint (as it should be invisible)—is naturally small. Thus, watermarks become volatile and can be interfered with via minute perturbations.

Perceptual Loss. *Learned Perceptual Losses (LPIPS):* They model human vision [291]. Such losses are pretrained on different datasets and have varying architectures, which affects

their effectiveness. After experimenting with several candidates, we found *DeepVGGLoss-LPIPS* (*DVL*) [57] for large images ($\geqslant 256\times256$) and *LPIPS-Alex* (*Alex*) [291] otherwise work best, regardless of the scheme. High-frequency modifications may cause sporadic peaks to appear. *Alex* fails to fathom the fragility of the larger images' higher resolution, often overlooking dominant, visible singleton peaks, while *DVL* better captures this phenomenon. As these images are larger, many invisible, smaller violations distributed throughout them sufficiently affect the spectrum, and *DVL* enables us to introduce such perturbations. For smaller images, *DVL* is too restrictive, prioritizing less destructive disruptions despite more powerful alternatives being applicable because of the reduced resolution, which can mask slightly stronger interferences. We find *Alex* in general better suited for realizing such changes. In contrast, due to the low dimensionality, *DVL*'s permitted small perturbations will not suffice, and increasing its budget quickly results in visible cues as *DVL* cannot distinguish between these nuances at this resolution. This problem is absent in low-frequency optimizations that are of a different nature (see §4.5.2), where *DVL* can be used universally.

Norm: To further ensure no such peaks appear (especially for low resolutions), we use the $\|\|_2$ norm to place a geometric bound on the distance from the watermarked image, allowing us to slightly increase the budget used for *LPIPS*. Whenever *LPIPS* risks introducing such peaks, $\|\|_2$ grows, forcing the procedure to explore different directions. Yet, $\|\|_2$ alone is insufficient since geometric constraints cannot capture visual similarity and will either lead to sub-optimal solutions if used with extremely small thresholds or impact quality otherwise. This is especially true since we operate in a *black-box* framework without model feedback, potentially requiring larger perturbations, entailing that the perceptual loss should be accurate to preserve quality.

Final High-Frequency Destruction Procedure. *(for non-semantic watermarks)* Using the above components, we implement *eq.* (4.1). We only need to identify the rules for updating the sample to define our search space. Since we seek high-frequency disruptions that affect the rates at which *individual* pixels vary w.r.t. their neighbors, we should modify each pixel directly. Thus, our algorithm accepts watermarked image $\boldsymbol{x}_w$, *LPIPS* loss ℓ_p, *LPIPS* threshold t_{ℓ_p}, and $\|\|_2$ threshold $t_{\|\|_2}$, and finds modifier $\boldsymbol{\delta}$ that maximizes *DFL* as follows:

$$\boldsymbol{x}_{nw} = \boldsymbol{x}_w + \underset{\boldsymbol{\delta}}{argmin} \left[\begin{array}{c} c_{\ell_p} \cdot ReLU_{\ell_p}(\boldsymbol{x}_w + \boldsymbol{\delta},\ \boldsymbol{x}_w) + c_{\|\|_2} \cdot ReLU_{\|\|_2}(\boldsymbol{x}_w + \boldsymbol{\delta},\ \boldsymbol{x}_w) \\ -DFL(\boldsymbol{x}_w + \boldsymbol{\delta},\ \boldsymbol{x}_w) \end{array} \right] \tag{4.2}$$

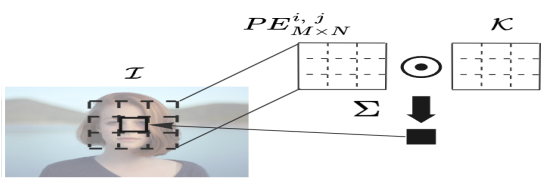
$ReLU_\ell(\boldsymbol{x},\ \boldsymbol{v})$ denotes $max\{\ell(\boldsymbol{x},\ \boldsymbol{v}) - t_\ell, 0\}$ for any loss ℓ. Effectively, we instantiate ℓ_d from *eq.* (4.1) with two components: *LPIPS* loss ℓ_p and norm $\|\|_2$, each with its threshold, while *DFL* is the spectral loss. The vector $\boldsymbol{\delta}$ captures the changes to $\boldsymbol{x}_w$ to generate the

non-watermarked $\boldsymbol{x}_{nw} = \boldsymbol{x}_w + \boldsymbol{\delta}$. We minimize -*DFL*, maximizing *DFL*. The constants c_{ℓ_p} and $c_{\|\|_2}$ balance the spectral and perceptual losses. We enforce that each perceptual loss (ℓ_d) component (ℓ_p and $\|\|_2$) remains below its threshold (t_{ℓ_p} or $t_{\|\|_2}$) through the term $ReLU_{\ell_d}(\boldsymbol{x}_w + \boldsymbol{\delta},\ \boldsymbol{x}_w)$: When ℓ_d exceeds t_{ℓ_d}, it evaluates to $\ell_d(\boldsymbol{x}_w + \boldsymbol{\delta},\ \boldsymbol{x}_w) - t_{\ell_d}$, causing ℓ_d's gradients to produce a sample closer to $\boldsymbol{x}_w$ in the next iteration. Otherwise, it returns 0, discarding ℓ_d's gradients and maximizing *DFL* unconditionally. $c_{\|\|_2}$ is a fixed *large* coefficient that strongly prevents the abovementioned peaks. c_{ℓ_p} is determined via a binary search: We start with a large c_{ℓ_p}, assigning a larger weight to ℓ_p's gradients and run the optimization for several steps. If the perceptual threshold is satisfied by the found perturbation, c_{ℓ_p} is decreased, increasing *DFL*'s dominance at the expense of larger visual deviations. Otherwise, c_{ℓ_p} is amplified. We return the found sample that best maximizes *DFL* while meeting the similarity constraints. See Appendix B.1 for details.

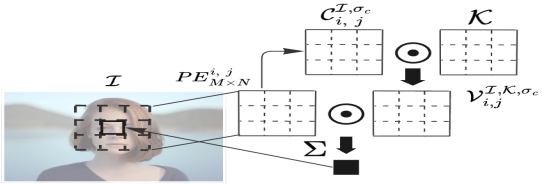
4.5.2 Attacking Semantic Watermarks

Semantic watermarks are in low frequencies, making them difficult to address. In addition to their large magnitudes (see §4.4.1), low frequencies correspond to many neighboring pixels varying gradually, indicating these watermarks are rooted in the image's texture. Thus, removing them requires *systemic* and structural changes. Direct optimizations similar to §4.5.1 fail since they operate on the pixels and, at each step, individual pixels are updated in different directions. This targets higher frequencies, introducing significant changes yet ***only*** at a few points, failing to progress due to the perceptual constraints, and consequently incurring minimal changes at low frequencies. We could limit the procedure to optimize *DFL* over low-frequencies only. Yet, as significant changes are required in this range, without a methodical scheme, we will converge to a sub-optimal solution wherein a certain region receives substantial modifications, leading the perceptual loss to prevent further updates. This is because the process fails to account for inter-pixel links, and *DFL* does not spread the changes over the image.

We develop a visually-aware attack inspired by image filtering. While traditional filters apply systemic spectral disruptions, they fail to remove watermarks due to their content-agnostic nature, limiting the distortions they may introduce without leaving crucial visual cues. Our adversarial alternative retains quality by learning the filters' weights to solve an optimization problem that maximizes low-frequency disruptions, resorting to a perceptual loss to preserve critical information. First, we motivate our design through guiding principles. We then provide technical details and present our optimizable filters. Finally, we introduce our attack.



(a) Basic: To calculate the output at $(i,\ j)$, the patch $PE^{i,j}_{M\times N}$ (where $M\times N$ is the kernel size) around $(i,\ j)$ is element-wise multiplied by the *spectral kernel* $\boldsymbol{\mathcal{K}}$, summing over this product.



(b) Guided: The *color kernel* $\boldsymbol{\mathcal{C}}^{\boldsymbol{\mathcal{I}},\sigma_c}_{i,j}$ is first computed, assigning a modulation factor to each neighbor in the vicinity $PE^{i,j}_{M\times N}$ of $(i,\ j)$. These are element-wise multiplied by the *spectral kernel* $\boldsymbol{\mathcal{K}}$ to limit the contribution from neighbors whose values differ significantly from $(i,\ j)$'s. This yields the *effective kernel* $\boldsymbol{\mathcal{V}}^{\boldsymbol{\mathcal{I}},\boldsymbol{\mathcal{K}},\sigma_c}_{i,j}$ used to calculate the output at $(i,\ j)$ similar to basic filters.

Figure 4.3: Operation of known filters.

Motivation

Basic Filtering: An image is convolved with a filter ($M\times N$ *spectral kernel*), moved s.t. its center is aligned with a single pixel $(i,\ j)$ at each step. The output at $(i,\ j)$ is calculated by multiplying the kernel by the pixel's vicinity and summing over the product— see Fig. 4.3(a). The kernel's values are non-negative and sum to 1, making this output an averaged sum of the pixel and its neighbors. This performs noise reduction, decaying unwanted peaks. The kernel's elements usually grow toward the center, assigning a larger weight to the original pixel to preserve the content.

Convolution in space translates to multiplication in the frequency domain: Given the spectra of the image and filter, it produces an image with a spectrum equivalent to their product. Thus, filters are good candidates for spectral manipulations. Yet, the outputs at any pixel are calculated using the same kernel. Hence, their potential to remove watermarks is limited [164, 298] due to their content-agnostic nature, making them only applicable with mild parameters as stronger blurring impacts quality.

Guided Filtering: These advanced filters tackle the information loss under stronger distortions, allowing more effective noise reduction while preserving crucial details. They

are edge-preserving, minimizing the disruptions at these critical boundary points between different objects. Instead of using a uniform kernel, they involve an additional per-pixel *color kernel* that weakens the filter at edge points based on geometric principles: If some values in the pixel's vicinity drastically differ from it, then it lies on an edge, as these neighbors belong to other components and the *spectral kernel*'s strength here must be diminished to preserve defining information. The *color kernel* computes a modulation factor for every neighbor in the filter's vicinity around $(i,\ j)$ based on its value's difference from $(i,\ j)$'s. The *effective kernel* for calculating $(i,\ j)$'s output is the Hadamard product of the two kernels— see Fig. 4.3(b). Guided filters fail to remove watermarks despite their stronger blurring (see §4.6.2) since they merely follow geometric heuristics that still cannot fully capture human vision or target specific image characteristics, limiting the allowed distortion. In practice, less *visually-critical* areas can even be changed more drastically, regardless.

Adversarial Filtering (Our Attack): Based on the above, structure-aware filters can better maximize the spectral distortion with minimal impact on crucial regions. Similar to guided filters, we design a process for learning a set of per-pixel kernels. Yet, instead of relying on geometric heuristics, our method learns the kernels' parameters s.t. they retain ***perceptual similarity*** to the original image from a human's perspective, allowing for stronger filtering.

Formalizing Filters

Given image $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{Q\times R}$, $PE^{i,j}_{M\times N} : \mathbb{R}^{Q\times R} \longrightarrow \mathbb{R}^{M\times N}$ denotes the *local patch extractor* that outputs the $M\times N$ vicinity around its $(i,\ j)^{th}$ pixel s.t. this pixel is at the center (i.e., $PE^{i,j}_{M\times N}(\boldsymbol{\mathcal{I}})(\lfloor\frac{M}{2}\rfloor,\ \lfloor\frac{N}{2}\rfloor) = \boldsymbol{\mathcal{I}}(i,\ j)$)— see Fig. 4.3(a). The *global patch extractor* $PE_{M\times N} : \mathbb{R}^{Q\times R} \longrightarrow \mathbb{R}^{Q\times R\times M\times N}$ outputs a matrix that stores in cell $(i,\ j)$ the *local patch* $PE^{i,j}_{M\times N}$. That is, $PE_{M\times N}(\boldsymbol{\mathcal{I}})$'s $(i,\ j)^{th}$ cell contains the $M\times N$ patch around $(i,\ j)$ in $\boldsymbol{\mathcal{I}}$. We formalize filters to present our solution.

Basic Filtering: Basic filters simply apply the known convolution with a fixed *spectral kernel* $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{M\times N}$. Denoting element-wise multiplication as $\odot$, these filters can be written as $\mathcal{F}_{\boldsymbol{\mathcal{K}}} : \mathbb{R}^{Q\times R} \longrightarrow \mathbb{R}^{Q\times R}$ s.t. $\mathcal{F}_{\boldsymbol{\mathcal{K}}}(\boldsymbol{\mathcal{I}})(i,\ j) = \sum_{m,n} PE^{i,j}_{M\times N}(\boldsymbol{\mathcal{I}}) \odot \boldsymbol{\mathcal{K}}$ (see Fig. 4.3(a)).

Guided Filtering: These employ additional per-pixel *color kernel*s to modulate distortions at critical points (Fig. 4.3(b)). Given kernel size $M\times N$, the *color kernel* for each $(i,\ j)$ is the matrix whose $(m,\ n)$ cell is $\boldsymbol{\mathcal{C}}^{\boldsymbol{\mathcal{I}},\sigma_c}_{i,j}(m,\ n) = e^{-\frac{\|\boldsymbol{\mathcal{I}}(i+m-\lfloor\frac{M}{2}\rfloor,\ j+n-\lfloor\frac{N}{2}\rfloor)-\boldsymbol{\mathcal{I}}(i,\ j)\|_1}{2\sigma_c^2}}$. σ_c is a permissiveness controlling the modulation intensity. The $\|\|_1$ norm accounts for each pixel potentially having multiple color channels. For instance, *RGB* images associate each

pixel with three values. The *effective kernel* for computing the output at $(i\ j)$ is then $\boldsymbol{\mathcal{V}}_{i,j}^{\boldsymbol{\mathcal{I}},\boldsymbol{\mathcal{K}},\sigma_c} = \frac{\boldsymbol{\mathcal{C}}_{i,j}^{\boldsymbol{\mathcal{I}},\sigma_c} \odot \boldsymbol{\mathcal{K}}}{\sum_{m,n} \boldsymbol{\mathcal{C}}_{i,j}^{\boldsymbol{\mathcal{I}},\sigma_c} \odot \boldsymbol{\mathcal{K}}}$. At the center, $\boldsymbol{\mathcal{C}}_{i,j}^{\boldsymbol{\mathcal{I}},\sigma_c}(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{N}{2} \rfloor) = 1$, indicating the contribution of the original $\boldsymbol{\mathcal{I}}(i,\ j)$ is not decayed. For neighbors, large distances decrease their coefficients and limit their contributions. If we define $\boldsymbol{\mathcal{V}}^{\boldsymbol{\mathcal{I}},\boldsymbol{\mathcal{K}},\sigma_c}$ as the matrix whose $(i,\ j)^{th}$ cell contains $\mathcal{V}_{i,j}^{\boldsymbol{\mathcal{I}},\boldsymbol{\mathcal{K}},\sigma_c}$, the guided filter is given as $GF_{\boldsymbol{\mathcal{K}},\sigma_c}(\boldsymbol{\mathcal{I}}) = \sum_{m,n} PE_{M\times N}(\boldsymbol{\mathcal{I}}) \odot \boldsymbol{\mathcal{V}}^{\boldsymbol{\mathcal{I}},\boldsymbol{\mathcal{K}},\sigma_c}$.

Optimizable Filters (Ours): Guided filters' drawback is that they employ the same *spectral kernel* $\boldsymbol{\mathcal{K}}$ for all pixels, changing only the *color kernel* computed via the same formula at all locations. Instead, we allow each pixel $(i,\ j)$ to have a unique *spectral kernel* $\boldsymbol{\mathcal{K}}_{i,j}$, with weights that can differ significantly between locations based on feedback from a perceptual loss that determines their visual importance and allows maximizing the destruction at less critical regions. Our filters consist of a set $\boldsymbol{\mathcal{K}}^*$ of local $\boldsymbol{\mathcal{K}}_{i,j}$'s. We include *color kernel*s to avoid exploring unwanted directions that violate vital geometric constraints. While these may be limiting, ignoring them causes unnatural distortions if the differences between neighboring pixels are extremely large, making them essential for remedying potential perceptual loss imperfections. When the distances are reasonably large but satisfy the perceptual loss with high confidence, it will assign larger weights that counteract the *color kernel*. Only when these distances are enormous do they become prohibitive, which is ideal. Our optimizable filter is:

$$\forall \boldsymbol{\mathcal{I}}:\ OF_{\boldsymbol{\mathcal{K}}^*,\boldsymbol{\mathcal{R}},\sigma_c}(\boldsymbol{\mathcal{I}}) = \sum_{m,n} PE_{M\times N}(\boldsymbol{\mathcal{I}}) \odot \boldsymbol{\mathcal{V}}^{\boldsymbol{\mathcal{R}},\boldsymbol{\mathcal{K}}^*,\sigma_c}$$

$\boldsymbol{\mathcal{R}}$ is a reference image for obtaining the *color kernel*s, set to $\boldsymbol{\mathcal{R}} \equiv \boldsymbol{x}_w$. We now show how to learn $\boldsymbol{\mathcal{K}}^*$'s weights.

Adversarial Filtering

Let $\{t \in [\![T]\!] | OF_{\boldsymbol{\mathcal{K}}_t^*,\boldsymbol{x}_w,\sigma_{c_t}}\}$ be a series of T optimizable filters (with *color kernel*s computed using $\boldsymbol{x}_w$), each with their permissiveness and size. We define the composition $\prod_{\boldsymbol{OF}}^{T} \equiv OF_{\boldsymbol{\mathcal{K}}_1^*,\boldsymbol{x}_w,\sigma_{c_1}} \circ OF_{\boldsymbol{\mathcal{K}}_2^*,\boldsymbol{x}_w,\sigma_{c_2}} \circ \cdots \circ OF_{\boldsymbol{\mathcal{K}}_T^*,\boldsymbol{x}_w,\sigma_{c_T}}$. Let ℓ_d be a perceptual loss, t_{ℓ_d} a similarity threshold, and ℓ_F a low-frequency loss. We instantiate *eq.* (4.1) to maximize ℓ_F as:

$$\begin{aligned} \max_{\{\theta_{\boldsymbol{\mathcal{K}}_t^*}\},\delta} & \left[\ell_F(\boldsymbol{x}_w,\ \prod_{\boldsymbol{OF}}^{T}(\boldsymbol{x}_w + \boldsymbol{\delta}))\right] \\ & s.t.\ \ell_d(\boldsymbol{x}_w,\ \prod_{\boldsymbol{OF}}^{T}(\boldsymbol{x}_w + \boldsymbol{\delta}))) \leqslant t_{\ell_d} \end{aligned} \tag{4.3}$$

where $\theta_{\mathcal{K}_t^*}$ are the kernels' weights and $\boldsymbol{\delta}$ is a modifier that directly optimizes $\boldsymbol{x}_w$, similar to *eq.* (4.2). We maximize the spectral loss by sequentially propagating the input through a network of filters. Recall that direct modifications are not effective for disrupting low frequencies in the absence of filters due to their inclination to cause high-frequency disturbances rather than systemic shifts in the values of many neighboring pixels in a gradual manner— see §4.5.2. Thanks to their convolutional nature, this no longer holds with our filters, making their outputs combinations of neighboring pixels. We still include $\boldsymbol{\delta}$ that directly manipulates the input despite its inefficacy in the absence of filters. This is because the originally high-frequency disturbances caused by such modifications will now be distributed over neighbors of the receiving pixels due to our filters, translating them into useful low-frequency noise, causing further disruptions beyond the filters' independent ability. We employ multiple (T) filters stacked after one another. The reason is that chaining several filters with various kernel sizes (geometries) targets different directions and slopes where the watermark could have been injected since watermarking is content-dependent and not all images can be changed similarly.

Spectral Loss

The former *DFL* is unsuitable for low frequencies (see §4.5.2) as it fails to force the changes to spread throughout the image. The collective energy at higher frequencies is much smaller in comparison, significantly reducing the total number of pixels that need to change to erase watermarks in these bands, which is achievable without violating the perceptual constraints. In contrast, sufficiently manipulating lower frequencies can rapidly change the structure, hindering convergence to an optimal solution if the distortion is not carefully distributed. This loss should also ensure our kernels perform the filtering operation, which *DFL* fails to enforce. Better alternatives are:

Filter Regularization Loss (FRL): It enforces that our kernels indeed operate as (smoothing) filters and is given as:

$$\forall \boldsymbol{x} \in \mathbb{R}^{Q \times R} : \; FRL(\boldsymbol{x}) = -\sum_{t \in [\![T]\!]} FRL_{\boldsymbol{t}}(\boldsymbol{x})$$
$$s.t.\ \forall t : \quad FRL_{\boldsymbol{t}}(\boldsymbol{x}) = \|\boldsymbol{x} - \overset{M_t \times N_t}{\boldsymbol{\mu}_{\frac{1}{2}}}(\boldsymbol{x})\|_2$$

$M_t \times N_t$ is the dimensionality of the kernels in our t^{th} set $\boldsymbol{\mathcal{K}}_t^*$ and $\overset{M_t \times N_t}{\boldsymbol{\mu}_{\frac{1}{2}}}$ computes the median value in this vicinity around each pixel. Maximizing *FRL* minimizes $FRL_{\boldsymbol{t}}$ for every filter t, which corresponds to all pixels' distances from the median in the filter's environment

around them, bringing pixels closer to these local medians. This eliminates variations to the extent possible under the perceptual loss, leading to smoother outputs. As the watermark is distributed over the image, such patches would exhibit gradual "watermarking" variations that will now be lost. A major benefit of *FRL* is output consistency.

Mean Pool Loss (MPL): FRL is insufficient, ensuring only proximity of the pixels to the local medians but imposing no constraints on these medians. The *means* of the various patches in the *watermarked* image are highly influenced by the watermark that causes many pixels to change their values. Due to the perceptual loss that strives to retain information, the procedure may converge to a sub-optimal solution, skewing these medians toward the original *means* and only slightly affecting pixel intensities, retaining many shifts associated with the watermark. Thus, we require *MPL*:

$$\forall \boldsymbol{x}, \boldsymbol{v} \in \mathbb{R}^{Q \times R}: \quad MPL^{L \times P}(\boldsymbol{x},\ \boldsymbol{v}) = \|\overline{\boldsymbol{x}}^{L \times P} - \overline{\boldsymbol{v}}^{L \times P}\|_1$$

$\overline{\boldsymbol{x}}^{L \times P}$ denotes mean pooling with an $L \times P$ window. *MPL* takes the means over each $L \times P$ patch in both inputs and calculates the ℓ_1 difference. As most pixels do not encode watermark information, maximizing the means' differences (w.r.t. the watermarked image) while demanding proximity to the medians (using *FRL*) primarily affects pixels located far (those that potentially encode the watermark) from all others, bringing them closer to the rest and eliminating watermarking variations. A proper low-frequency loss must spread the changes over multiple regions (see §4.5.2). *MPL* does so by dividing the image into patches and demanding that each receive a portion of the change, implying the number of patches must be large. Yet, each must contain enough pixels to manipulate *low* frequencies. We choose $MPL^{5\times5}$ as a compromise, balancing these requirements. Calculating *MPL* for all filter sizes similar to *FRL* is less desirable as when the sizes shift away from this sweet spot, one of the conditions is violated. Note that, without *FRL*, *MPL* is insufficient as it can lead to all pixels in each patch receiving a similar modification ϵ, simply changing its brightness. This satisfies the perceptual constraints but leaves the spectrum unaffected as the pixel variation rates remain identical.

Final Low-Frequency Destruction Procedure *(for semantic watermarks)*

We replace *eq.* (4.3)'s abstractions with the above components and $\ell_d = DVL$ as the perceptual loss (see §4.5.1). We also use $\|\|_2$ regularization to avoid drastic updates and encourage low-frequency, gradual changes. Yet, it is multiplied by a small coefficient to only regulate the process without further influencing the outputs, leaving this primarily to

DVL. Defining $\boldsymbol{x}_w^{OF_T,\boldsymbol{\delta}} \equiv \prod_{\boldsymbol{OF}}^{T}(\boldsymbol{x}_w + \boldsymbol{\delta})$, we have:

$$\min_{\{\theta_{\boldsymbol{\kappa}_t^*}\},\boldsymbol{\delta}} \begin{bmatrix} c_{DVL} \cdot ReLU_{DVL}(\boldsymbol{x}_w,\ \boldsymbol{x}_w^{OF_T,\boldsymbol{\delta}}) + \lambda_{\|\|_2} \cdot \|\boldsymbol{x}_w^{OF_T,\boldsymbol{\delta}} - \boldsymbol{x}_w\|_2 \\ -MPL^{5\times5}(\boldsymbol{x}_w,\ \boldsymbol{x}_w^{OF_T,\boldsymbol{\delta}}) - \lambda_{FRL} \cdot FRL(\boldsymbol{x}_w^{OF_T,\boldsymbol{\delta}}) \end{bmatrix} \tag{4.4}$$

We maximize the loss $\ell_F = MPL^{5\times5} + \lambda_{FRL} \cdot FRL$ by minimizing its negation similar to §4.5.1. $ReLU_{DVL}$ ensures similarity, keeping the *DVL* distance from $\boldsymbol{x}_w$ below t_{DVL}. The resulting changes are gradual due to our filters, whose weights are learned to achieve this behavior thanks to our spectral loss, yielding optimal $\{\widehat{\theta}_{\boldsymbol{\kappa}_t^*}\}, \hat{\boldsymbol{\delta}}$ s.t. the final solution is $\boldsymbol{x}_{nw} = \boldsymbol{x}_w^{\widehat{OF}_T,\hat{\boldsymbol{\delta}}}$, where $\widehat{OF}_T$ denotes the filters applied using $\{\widehat{\theta}_{\boldsymbol{\kappa}_t^*}\}$. As each kernel's values should be non-negative and sum to 1, we apply *Softmax* to the learned weights to obtain effective values in this range. λ_{FRL} balances *FRL* and *MPL* and is calibrated s.t. both are optimized, while $\lambda_{\|\|_2}$ limits the impact of the $\|\|_2$ regularization. c_{DVL} balances the spectral and visual losses and is determined via a binary search, as in §4.5.1. We found a single filter network architecture universally effective. Details are in Appendix B.1.

Discussion: Without our filters, the modifications at each step are less systemic, allowing different pixels to receive significant updates in opposing directions. Thus, optimization will converge to a sub-optimal solution since massive changes that a *subset* of pixels receive will substantially increase the visual loss, preventing further progress. While our low-frequency loss discourages this behavior and encourages uniform updates, it remains lacking, constantly countered by the visual loss that strives to retain similarity, limiting the number of pixels allowed to change their values, deeming such structural modifications less desirable compared to high-frequency changes that affect discrete pixels thereby remaining potentially unnoticeable. Our filters **force** prioritizing collective updates, drastically improving performance. Results establishing these benefits are in §4.6.2.

While effective against *semantic* watermarks, this attack is unsuitable for *non-semantic* ones (see §4.6.2); The low-frequency loss and filters encourage low-magnitude, yet collective, changes and prevent significant updates to specific pixels (as opposed to §4.5.1). Yet, *non-semantic* watermarks are injected in high frequencies and thus rely on large variations between neighboring pixels. As this process can only slightly affect such peaks, the watermark is preserved.

4.6 Experiments

4.6.1 Setup

Schemes. We consider the *SOTA* watermarking schemes in Table 4.1, covering all categories from §4.2, selected for appearing in top venues, with several remaining unbroken by a realistic *query-free, black-box* attacker. Among the included *generator-specific* schemes, *Yu1* [282], *Yu2* [281], and *PTW* [164] watermark *Generative Adversarial Networks (GANS)* [86], while *StableSignature* [78] and *TRW* [260] operate on *LDMs* [211]. We are not aware of *semantic* schemes from top venues other than those we consider, which are the focus of previous work [298, 214, 15]. As most recent systems are *generator-specific*, we include five such schemes and two *general-purpose* ones. Compared to existing work (see §4.2.2), we span a wider variety of recent systems.

	General-Purpose	***Generator-Specific***
Semantic	*StegaStamp [236]*	*TRW [260]*
Non-Semantic	*HiDDeN [303]*	*Yu1 [282], Yu2 [281], PTW [164], StableSignature [78]*

Table 4.1: Studied *SOTA* watermarking schemes.

Data. Most *generator-specific* schemes require no input data. Still, they were trained on various datasets: *TRW* and *StableSignature* on *LAION-5B* [219], *Yu1* and *Yu2* on *CelebA* [125], and *PTW* on *FFHQ* [126]. As *TRW* and *StableSignature* watermark text-to-image *LDM*s (*StableDiffusion* [46]), we use *StableDiffusion*'s prompts. For *HiDDeN*, we take samples from *COCO* [158] on which it was trained. *StegaStamp* watermarks 400×400 images. Thus, we downscale *CelebA-HQ-1024* samples[2]. We evaluate each system on 100 random images, similar to previous work [120].

Metrics. Following §4.3, we are concerned with two metrics:

1) Detection: Bit Accuracy. For binary watermarks, detection occurs if the ratio of bits matching the watermark in the sequence extracted from the image is above some threshold t_w. As in previous work [298, 165], we select t_w that can reject the null hypothesis H_0 that k or more matching bits were randomly extracted with $p < 0.01$. This has probability $P(X \geqslant k|H_0) = \sum_{i=k}^{n} \binom{n}{k} \frac{1}{2^n}$, for watermark length n. Thus, $t_w = \frac{k}{n}$ for which this probability is below 0.01 is selected. We denote the portion of successfully extracted bits as the *bit accuracy*, which we compare against t_w. Watermark lengths are in Appendix B.2.

[2] *FFHQ-1024* samples yield similar results (excluded for brevity).

Inverse Distance. *TRW*'s watermarks are non-binary and unbounded, and detection occurs if the ℓ_1 distance (effectively, the *Mean Absolute Error— MAE*) of the extracted signal from the actual watermark is below a threshold determined s.t. the ratio of falsely detected non-watermarked images (*FPR*) is low. We choose a higher *FPR* (0.02) than the original paper [260] (0.01), making the attacker's task harder, obtaining $MAE = 71$. For consistency, we use this distance's *inverse*: for the watermark to be detected, the *MAE*'s *inverse* must be $\geqslant t_w = \frac{1}{71}$.

Security Advantage. This *binary* indicator is the scheme's ability to detect over 50% of *watermarked* images. Most genuine content is not controversial, and detectors will not be invoked to determine its authenticity. They are only necessary when suspicious content surfaces. Absent additional knowledge, we assume a natural-looking controversial image has equal probabilities of being real or fake. Here, the best *trivial* detector that achieves the highest average accuracy on suspicious images would be a random classifier. An attacker will have a 50% chance of fooling it with a fake image. For a scheme to have a *Security Advantage*, it must detect *watermarked* images at a higher rate. Otherwise, the resources invested in it will be in vain as it offers no additional protection against attackers compared to random systems. While such a scheme will perform better on non-watermarked images as it is trained to detect the existence of a watermark, this is less relevant from a security aspect where only identifying the fake (watermarked) content matters.

2) Quality: We use two common [298, 15, 165, 110, 164] human vision-inspired metrics: *LPIPS* [291] and *Frèchet Inception Distance (FID)* [102]. For fairness, we avoid *LPIPS* methods *UnMarker* explicitly optimizes (see §4.5.1), resorting to *LPIPS-VGG* [291]. *FID* compares the attack samples' distribution to that of watermarked images, measuring *dataset-wise* similarity. We exclude geometric metrics (*PSNR* or norms) as we find them misleading. For instance, although visual metrics (Table 4.4) and human inspection (Appendix B.4) verify the *VAEAttack*'s inferior quality, geometric methods deem it comparable with *UnMarker*.

Basic Manipulations. We evaluate *Center Cropping (*10%*)*, *JPEG*, *Quantization*, and *Gaussian Blurring* (filtering). Their parameters (see Appendix B.3) were chosen following previous work [164, 298] and to allow maximum distortion while retaining acceptable quality. We also evaluate *Guided Blurring*, showing how even these stronger filters cannot break robust watermarks, as argued in §4.5.2. To better explore these advanced filters, we even evaluate them with overly destructive parameters that do not represent viable attacks.

SOTA Attacks. We compare *UnMarker* against *SOTA* watermark removal attacks: The *DiffusionAttack* [298, 214, 15] and *VAEAttack* [298, 15]. To our knowledge, these regeneration methods are the only practical (*query-free* and *black-box*) and effective attacks,

as adversarial attacks are disqualified for their unrealistic assumptions (see §4.2.2).

4.6.2 Evaluations

Watermark Utility

We ensure the chosen schemes' utility, following the requirements from §4.4.1: *Applicability*, *Stealthiness*, and *Robustness to Manipulations*. While the latter demands withstanding basic manipulations, true robustness is contingent on resisting advanced attacks as well. We separate these cases for consistency with watermark papers that mainly consider basic manipulations. Advanced attacks are in §4.6.2. We focus on *Applicability* and *Robustness* as we use original implementations of *SOTA* systems whose *Stealthiness* has been shown by their authors.

Visual metrics are excluded here, as evaluating the naive manipulations' output quality is not our objective, knowing that our *basic* settings are quality-preserving [298, 164]. We do not consider rotation manipulations, following Zhao et al.'s [298] arguments w.r.t. their impracticality as they can be easily spotted and corrected before invoking the detector. Still, their finding of some robust schemes vulnerable to rotations may seem at odds with our assertions that these watermarks are in the *collective* spectral amplitudes, which should leave them unaffected by rotations. Yet, schemes are generally not ***explicitly designed*** to construct watermarks in the spectral amplitudes. Rather, this behavior is adopted when trained for robustness (see §4.4). Failure to expose these systems to a wider set of demands (e.g., resistance to rotations) may result in them mistakenly associating ***additional*** indicators, such as a specific orientation, with the watermark. As rotations are impractical, this is not an issue.

A detailed analysis is in Appendix B.3 alongside the results (Table B.2). All schemes detect watermarked images without manipulations: For binary watermarks, all *bit accuracies* approach 100%. For *TRW*, the *inverse distance* is 0.022, well over the 0.0141 threshold. All systems are fairly resistant to manipulations. *Semantic* schemes remain almost unaffected. Among *non-semantic* schemes, *Yu1* and *StableSignature* are the most resistant, retaining robustness against all *realistic* configurations. The latter shows some vulnerability to *Guided Blurring* with overly destructive parameters, but since this is not a practical attack, it remains reliable. *PTW* rejects most manipulations but is affected by reasonable *Quantization* (10), while the less advanced *HiDDeN* is vulnerable to *JPEG* and *Blurring*. *Yu2* resists manipulations except *cropping*, indicating high dependency on spatial attributes, making it unsuitable for security-critical deepfake detection by nature, as it violates the *robustness* requirement from §4.4.1.

The results demonstrate how all schemes except *Yu2* are robust to spatial manipulations (e.g., *cropping*), suggesting their watermarks may be in a different channel, which we next verify is the spectral amplitudes as expected. This is readily somewhat evident for *PTW* and *HiDDeN* that exhibit vulnerability to *Quantization* and *JPEG*, which affect high frequencies, while the remaining schemes require the more advanced attacks we present below. *Yu2* highly depends on information in the background, suggesting its watermarks are not constructed as robust, collective quantities but rather in volatile spatial locations.

UnMarker

We now evaluate *UnMarker*'s watermark removal capabilities. In §4.6.2, we compare it to *SOTA* attacks and assess its ability to preserve visual similarity.

Configuration. Our analysis proves that a conventionally robust scheme (under spatial manipulations) must inject its watermarks in the frequency space. Schemes that are spatial in nature and rely less on spectral watermarking (e.g., *Yu2*) may hence evade *UnMarker*'s spectral optimizations as they are not designed to target these inherently fragile signals. Yet, equipping *UnMarker* with a simple *cropping* layer suffices to defeat such weaker systems. Moreover, as spectral watermarks distribute their modifications throughout the image to minimize visual cues (see §4.4), *cropping* can also slightly "weaken" those by eliminating partial modifications residing in the background. This allows using a decreased distortion budget for the optimization stages.

Thus, UnMarker's design is as follows: We apply mild *cropping*, followed by our high-frequency disruptions (§4.5.1). Finally, our filtering stage (§4.5.2) eliminates low-frequency traces. This yields a full attack that targets all *carrier*s, erasing all watermarks. Attackers can adjust the *cropping* ratio or location to restrict the output to the actual content. For our experiments, 10% *center cropping* sufficiently retains the content. The stages' order does not affect the results.

UnMarker's hyperparameters should be calibrated to maximize the spectral loss regardless of the system under attack, complying with our threat model. Hence, they are only affected by properties of the image, such as its size, which we found to be the only relevant factor. The hyperparameters are in Appendix B.1. The size-dependent parameters are merely the thresholds and coefficients. Meanwhile, the number of filters and their kernels are invariant, as by combining multiple filter sizes, a single architecture can defeat any *semantic* watermark (see §4.5.2).

Results. We report settings combining different stages of *UnMarker*, identifying crucial insights that establish our theory. These configurations are *CH*— the high-frequency

destructive process preceded by *cropping*, *CL*— the low-frequency optimization (with *cropping*), *HL*— the two optimizations without *cropping*, and *CHL*, denoting the full *UnMarker* with all three stages. Results are in Table 4.2. Following §4.6.1, we say that an attack defeats a scheme (alternatively, *removes its watermark*) if it causes it to lose its *Security Advantage* (i.e., detection below 50%).

Non-Semantic Schemes. *Robust non-semantic* schemes (w.r.t. known manipulations) must embed their watermarks in high frequencies. Hence, small disruptions to this part of the spectrum where the energy concentration is low suffices to remove them. Our results corroborate this as all such schemes break under *CH* that combines *cropping* and high-frequency disruptions. The detection rates of the *conventionally* robust algorithms (that resist *cropping*— all but *Yu2*) remain almost unaffected by *CL*, which primarily targets low frequencies where they do not operate. *Yu2* is defeated by *CH*, but *HL*'s results prove this is not due to our optimization since the scheme is insensitive to manipulations in both parts of the spectrum, indicating that *cropping* only is responsible for this outcome. This comes as no surprise since *Yu2* employs *spatial* watermarking that enables it to resist spectral optimizations but makes it vulnerable to *cropping*.

Schemes that process larger images (*PTW* and *StableSignature*) are hardly sensitive to *cropping* but vulnerable to high-frequency manipulations, shown by their robustness to *CL* and fragility against *HL*, bringing their detection to 6% and 9%. *Yu1* and *HiDDeN* that watermark smaller 128×128 images retain 100% detection against *HL*. Yet, their *bit accuracies* drop sharply from 98.42% and 99.16% to 73.11% and 72.28%. The cause is our high-frequency destruction since *CL* that combines *cropping* and low-frequency disruptions offers almost no advantages over *cropping*.

Thus, there is a connection between the image size, sensitivity to *cropping*, and vulnerability to high-frequency modifications, which is easy to explain: Spatial *robustness* and *stealthiness* require spreading the watermark across the image. Meanwhile, *non-semantic* watermarks are restricted to specific regions (i.e., edges) where they can exploit the high-frequency nature (see §4.4.1) to remain invisible. As the size shrinks, such areas become less prevalent in central parts, forcing the changes to be delegated to the background and making them more vulnerable to *cropping*. Accordingly, the portion of the change each region receives grows as well, making it difficult to eliminate these watermarks with minimal modifications, explaining why they retain *bit accuracies* above the threshold despite a significant decline without *cropping*. Regardless, *mild cropping* then becomes capable of eliminating sufficient footprints to further decrease these values below the thresholds, as seen for *CH* (*CHL*)—although *cropping* alone is not enough without our spectral disruptions.

Larger images enable schemes to better reject *cropping* since this new-found resolution

Attack \ Scheme		*Yu1* (61%)		*Yu2* (63%)		*HiDDeN* (73%)		*PTW* (70%)		*StableSig.* (69%)		*StegaStamp* (63%)		*TRW* (0.0141)	
		Bit Acc↓	Detect↓	Bit Acc↓	Detect↓	Bit Acc↓	Detect↓	Bit Acc↓	Detect↓	Bit Acc↓	Detect↓	Bit Acc↓	Detect↓	Inv. Dist.↓	Detect↓
None		98.42%	100%	99.71%	100%	99.16%	100%	97.96%	100%	98.97%	100%	99.95%	100%	0.022	100%
Cropping	(10%)	74.56%	100%	53.71%	**0%**	86.84%	100%	93.9%	100%	97.86%	100%	93.96%	100%	0.0155	100%
UnMarker	*CH*	59.04%	33%	53.06%	**0%**	58.34%	1%	63.11%	11%	53.08%	**4%**	94.87%	100%	0.0152	99%
	CL	73.16%	100%	54.62%	**0%**	84.21%	98%	95.21%	100%	97.6%	100%	61.8%	**43%**	0.0141	55%
	HL	73.11%	100%	99.23%	100%	72.28%	66%	61.53%	**6%**	54.59%	9%	64.64%	59%	0.016	98%
	CHL	59.08%	**31%**	53.03%	**0%**	58%	**0%**	62.75%	11%	52.82%	**4%**	61.49%	**43%**	0.014	**40%**

Note: Performance without attacks and under cropping is included for reference. The full attack is in the last row (CHL). Numbers in parentheses are the detection thresholds, while Detect is the portion of images meeting them. The best attacks are in bold font. All schemes lose their Security Advantage.

Table 4.2: Performance of watermarking schemes against each stage of *UnMarker*.

facilitates embedding the watermark in central areas, but the footprint in these regions must be smaller. Thus, minute disruptions throughout the image become more effective, explaining their inability to withstand high-frequency optimizations alone.

***UnMarker* (*CHL*) defeats all *non-semantic* schemes, with the most resistant being *Yu1*, which retains a detection rate of 31%. Yet, even *Yu1* still loses its Security Advantage (and so do all other schemes).**

Semantic Schemes. *StegaStamp* and *TRW* resist high-frequency disruptions (*CH*), with their detection fixed at 100% and the *bit accuracies* (*inverse distances*) retaining almost identical values to those observed for *cropping* alone. Yet, both are vulnerable to low-frequency disruptions, evident by their significant performance decline against *CL*: *StegaStamp*'s detection drops to 43% while *TRW*'s becomes 55%. This cements our arguments regarding *semantic* watermarks being primarily in lower frequencies and the soundness of our filtering approach. Comparing *CL* to the final results (*CHL*), we observe identical detection for *StegaStamp*. *TRW* exhibits an additional decline to 40%, indicating a small portion of its watermark is in high frequencies.

StegaStamp's watermarks are more visible (see §4.2.1), better engraving them in the *main* content, and increasing their robustness to spatial manipulations (*cropping*). Since *TRW* produces less visible watermarks despite targeting low frequencies where significant watermarking footprints are required, it behaves similar to *non-semantic* schemes for small images, forcing a considerable portion of these changes to populate the background. The results for *cropping*, *CL, CH* and *CHL* prove this; while *cropping* minimally affects *StegaStamp* (the *bit accuracy* only drops to 93.96%), *TRW* is more sensitive as its *inverse distance* decreases from 0.022 to 0.0155 when the threshold is only 0.0141. Although this is in the detection range, combining *cropping* with *UnMarker*'s low-frequency disruptions (*CL, CHL*) erases its watermarks (eliminates its *Security Advantage*).

Spectral optimizations (*HL*) drastically lower *TRW*'s *inverse distance* (to 0.016), but do not erase its watermarks. This is because low-frequency distortions affect many pixels simultaneously. Under a strict budget, the number of regions where these rates can be altered is limited compared to high-frequencies. *StegaStamp*, whose watermarks are centralized, requires modifying fewer areas, making *HL* successful without *cropping*. Yet, we verified that low-frequency disruptions with a higher budget or number of steps break *TRW* without *cropping*, but this may introduce artifacts unless carefully done. *Cropping* is as effective but less demanding, confirming a flaw in *TRW* that targets non-critical regions.

***UnMarker* defeats all semantic schemes due to irreconcilable trade-offs, as robustness to spatial manipulations incurs vulnerability to spectral disruptions and vice versa. Even when not bringing detection to 0%, it leaves a scheme useless, eliminating its *Security Advantage*.**

Remarks: As seen throughout this section, existing schemes predominantly concentrate their watermark footprint in either low- or high-frequency components, even if partial overlap may occur. While certain methods (e.g., *TRW*) exhibit signal across both regions, a dominant concentration remains observable in low frequencies. In practice, *UnMarker* supports both stages simultaneously (*CHL*) while preserving visual quality (see §4.6.2), enabling it to be applied without prior knowledge of the scheme's characteristics and allowing for effective removal even if the watermark footprint does not exhibit a dominant concentration in a single frequency range.

Effects of Adversarial Filtering

The reader may wonder whether the low-frequency loss alone can defeat *semantic* schemes, eliminating the need for our filters. Based on our arguments explaining how direct optimizations are of limited applicability against low-frequency watermarks due to their non-systemic nature (see §4.5.2), this is not the case. For validation, we compare *UnMarker (CHL)* against *semantic* schemes with and without our filters in the low-frequency attack stage (the two runs are otherwise identical). The results in Table 4.3 prove the filters necessary. While *TRW* experiences a slight degradation without filters, including them is far more effective, bringing its detection to 40% (compared to 86% otherwise). *StegaStamp* is unaffected when filters are excluded, retaining 100% detection. With filters, this drops to

Attack \ Scheme		*StegaStamp* (63%)				*TRW* (0.0141)			
		FID↓	*LPIPS*↓	Bit Acc↓	Detect↓	*FID*↓	*LPIPS*↓	Inv. Dist.↓	Detect↓
UnMarker	*with filters*	34.69	0.08	61.49%	43%	60.73	0.1	0.014	40%
	w/o filters	30.45	0.04	93.13%	100%	49.59	0.04	0.0147	86%

Table 4.3: *UnMarker*'s performance against *semantic* schemes with and without optimizable filters.

43%. We also verified the filter-free attack's ineffectiveness persists even if the perturbation slightly increases. Although similarity is slightly worsened by our filters as they encourage structural changes, the scores show that, even with filtering, *UnMarker* retains quality.

Comparison with *SOTA* attacks

We compare *UnMarker* to regeneration attacks— the *VAEAttack* [298, 15] and *DiffusionAttack* [298, 214, 15]. Despite being certifiably guaranteed to remove *non-semantic* watermarks, they fail against *semantic* methods. This robustness creates a false sense of security around *semantic* watermarks [298, 15], which *UnMarker* disproves. Additionally, they guarantee the removal of *non-semantic* watermarks only if the attack introduces sufficient noise to dilute the watermark, followed by a denoising process to restore the image's quality. Stronger watermarks require increased perturbations that prevent perfect reconstruction. Unnatural artifacts can be avoided by using advanced regeneration systems, explaining why the more sophisticated *DM*s outperform *VAE*s, as they can apply larger distortions while somewhat accurately retrieving the content. Yet, similarity to the original image may still be lost under such large budgets. *UnMarker* eliminates these hurdles, significantly decreasing the distortion while being effective even against the stronger *semantic* schemes.

As *UnMarker* uses *cropping*, we apply the *DiffusionAttack* with and without *cropping* for a fair comparison. The *VAEAttack*'s image quality was deemed inferior, disqualifying many of its results (see below), and thus a similar comparison was excluded. Results are in Table 4.4, and samples produced by the three attacks are in Appendix B.4.

Quality: The range for *LPIPS* is $[0, 1]$. Images with *LPIPS* distance at the 0.1 level (i.e., below 0.2) are visually similar [98]. Thus, we choose 0.15 as a cut-off beyond which we say the attack begins to cause visible changes. This does not mean that slightly higher scores imply severe degradation since the changes may integrate well with the content. Yet, scores far beyond this threshold attest to poor output nature. Despite *FID*'s known ability to measure quality, it has two major shortcomings: first, it compares the *datasets* of watermarked images and those that have undergone the attack and cannot judge samples on an image-by-image basis. It may also exhibit inconsistencies with human vision, making it less ideal in certain instances [159]. Thus, while *FID* is important for assessing quality, we prioritize *LPIPS*.

We do not consider the *VAEAttack* effective. It allows choosing from eight quality options that control the added noise before regeneration. We apply it with the three lowest options, as we found increasing this factor beyond 3 makes the attack almost always ineffective. Yet, all three options severely impact the quality, invalidating the attack. This

Attack \ Scheme		*Yu1* (61%)				*Yu2* (63%)				*HiDDeN* (73%)				*PTW* (70%)				*StableSignature* (69%)			
		FID↓	*LPIPS*↓	Bit Acc↓	Detect↓	*FID*↓	*LPIPS*↓	Bit Acc↓	Detect↓	*FID*↓	*LPIPS*↓	Bit Acc↓	Detect↓	*FID*↓	*LPIPS*↓	Bit Acc↓	Detect↓	*FID*↓	*LPIPS*↓	Bit Acc↓	Detect↓
None		*NA*	*NA*	98.42%	100%	*NA*	*NA*	99.71%	100%	*NA*	*NA*	99.16%	100%	*NA*	*NA*	97.96%	100%	*NA*	*NA*	98.97%	100%
UnMarker (CHL)		34.07	**0.15**	59.08%	31%	22.88	**0.1**	53.03%	**0%**	39.56	**0.08**	58%	**0%**	**11.9**	**0.14**	62.75%	11%	**30.87**	**0.05**	52.82%	4%
Diffusion Attack	*no cropping*	**12.38**	0.2	58.31%	28%	13.09	0.17	68.29%	87%	**34.1**	0.18	59.8%	3%	22.95	0.26	**55.53%**	7%	36.57	0.32	47.89%	**0%**
	+cropping	14.02	0.18	**54.24%**	8%	**11.98**	0.17	**51.26%**	3%	36.97	0.19	56.61%	**0%**	19.98	0.25	62.01%	16%	41.88	0.31	47.78%	**0%**
VAE Attack	*quality=3*	36.2	**0.15**	67.19%	97%	19.67	0.13	73.89%	96%	65.16	0.22	60.4%	8%	26.6	0.26	73.63%	89%	72.21	0.37	48.25%	**0%**
	quality=2	46.82	0.18	64.42%	81%	27.4	0.16	66.62%	78%	83.4	0.26	58.36%	1%	39.11	0.29	71.33%	79%	94.85	0.41	44.93%	**0%**
	quality=1	58.37	0.22	60.63%	59%	38.9	0.19	61%	36%	104.92	0.32	**56.26%**	2%	54.93	0.33	66.74%	36%	128.88	0.44	**43.87%**	**0%**

(a) Performance against non-semantic schemes.

Attack \ Scheme		*StegaStamp* (63%)				*TRW* (0.0141)			
		FID↓	*LPIPS*↓	Bit Acc↓	Detect↓	*FID*↓	*LPIPS*↓	Inv. Dist.↓	Detect↓
None		*NA*	*NA*	99.95%	100%	*NA*	*NA*	0.022	100%
UnMarker (CHL)		34.69	**0.08**	**61.49%**	**43%**	60.73	**0.1**	**0.014**	**40%**
Diffusion Attack	*no cropping*	33.15	0.36	83.21%	100%	41.69	0.32	0.0155	99%
	+cropping	**30.83**	0.33	77.84%	95%	**41.16**	0.31	0.0144	79%
VAE Attack	*quality=3*	54.89	0.42	97.9%	100%	71.91	0.38	0.0154	90%
	quality=2	70.86	0.46	94.7%	100%	93.56	0.41	0.015	88%
	quality=1	87.8	0.49	88.84%	100%	118.85	0.46	0.0145	67%

(b) Performance against semantic schemes.

Note: Performance in the absence of attacks is included for reference. For each metric, bold entries correspond to the best attack against the relevant scheme. Gray cells denote scores meeting our quality and removal criteria for LPIPS—0.15 and Detect— 50% (where the system loses its Security Advantage).

Table 4.4: Comparison with *SOTA* watermark removal attacks.

can be seen in Table 4.4, as both its *FID* and *LPIPS* are extremely worse (larger) than the other attacks whenever it degrades the detection rates. For larger images (over 256×256), even the best factor (3) is destructive, resulting in *LPIPS* scores at least 3.8× compared to *UnMarker* and well above the 0.15 cut-off. Its *FID*s are also inflated. While its *FID* against *TRW* (71.91) is close *UnMarker*'s (60.73), its *LPIPS* is much worse, indicating low quality. For smaller images, it retains similarity for *Yu1, Yu2* and *PTW* with *quality*=3, but fails to remove their watermarks. *HiDDeN*'s watermarks are removed with this factor at an acceptable *LPIPS* (0.22). Still, the failure against other systems limits this attack. Lowering the factor impacts quality, invalidating any results for larger images (including *PTW*). For *Yu1* and *Yu2*, as the attack fails to achieve remarkable removal for *quality*=2, the relevant case is *quality*=1. Both systems retain acceptable *LPIPS* scores (0.22 and 0.19) but violate the 0.15 threshold, and their *FID*s are far worse than the other attacks, proving the *VAEAttack* fails to preserve quality (see Appendix B.4).

The *DiffusionAttack* is controlled via a parameter t. Following Saberi et al. [214], $t>0.2$ causes significant deviations from the watermarked input. We found that for smaller images (128×128), this occurs at lower values. For these images, we set t=0.05, deeming it the turning point. A lower t results in less successful attacks. *UnMarker* achieves better

LPIPS ($\leqslant$0.15), ensuring similarity. The differences are drastic for larger images (over 256×256)— 0.1 at most for *UnMarker* compared to $\geqslant$0.37 for the *DiffusionAttack*. The higher *LPIPS* of *DiffusionAttack*, while not necessarily indicating significant deviations, entail that some nuances will appear, which may lead to its outputs being identified as fake— see Appendix B.4. The attacks' *FID*s alternate, occasionally outperforming one another. Yet, both attain far better *FID*s than the *VAEAttack*.

***UnMarker* attains the lowest *LPIPS*, with its worst being 0.15, ensuring resemblance to the watermarked images. The similarity between *UnMarker*'s *FID*s and the *DiffusionAttack*'s, which is known to produce high-quality outputs [298], further establishes its capabilities.**

Watermark Removal: Both *SOTA* attacks defeat *non-semantic* schemes, although the *VAEAttack*'s quality invalidates its results. *UnMarker* always outperforms this attack except against *StableSignature* for which *UnMarker* brings the detection to 4% while the *VAEAttack* reduces it to 0%. Still, this difference is insignificant. The *DiffusionAttack* and *UnMarker* have similar results on average, but the *DiffusionAttack* (+*cropping*) significantly outperforms *UnMarker* on *Yu1* (8% detection vs. 31%). Yet, as both eliminate its *Security Advantage*, specific numbers are of little relevance.

Our results validate regeneration's known inability to remove *semantic* watermarks [298, 214, 15]. The *VAEAttack* somewhat affects *TRW*, especially with quality 1, bringing its detection to 67%, but, as explained above, we consider it invalid. The *DiffusionAttack* leaves both detection rates almost unchanged despite reducing *StegaStamp*'s *bit accuracy* and *TRW*'s *inverse distance* from 99.95% and 0.022 to 83.21% and 0.0155. While *cropping* decreases these numbers to 77.84% and 0.0144 and makes the attack more successful against *TRW* (79% detection), *UnMarker* drastically outperforms it, reducing this to 40% and *StegaStamp*'s, against which *DiffusionAttack* is ineffective, to 43%.

Among the *SOTA* attacks, only the *DiffusionAttack* can remove watermarks without violating visual constraints. Yet, *UnMarker* is superior as it succeeds against *semantic* watermarks where the *DiffusionAttack* fails.

4.7 Discussion

4.7.1 Practicality

UnMarker is executed offline as it requires no detector interaction and takes up to 5 minutes with a *40GB A100 GPU*. Such machines are available for \$32.77/*hr* on *AWS*.

It requires no data, freeing it from the dependence on denoising pipelines (e.g., *DM*s). Regeneration attacks are affected by these systems' fidelity and the commonalities between their training data and the watermarked images, as we showed for the *VAEAttack*. While the *DiffusionAttack* achieves better quality (but worse than *UnMarker*'s), its *DM* and all studied schemes were trained on public data. Providers possess exclusive data that differ from those used to train the attacker's *DM*, making it likely to impact quality.

4.7.2 Mitigations

In the context of defensive watermarking, mitigations should allow recovery from removal attacks, enabling providers to continue using the scheme. *UnMarker* is an adversarial attack. Adversarial training (*AT*) [242] is a popular mitigation in *ML*, wherein the model (detector) is trained on malicious samples from known attacks so that it learns to classify them correctly. Mitigations can also target the model's inputs: the provider applies a pre-processing stage (e.g., purification [182]) to eliminate disruptions and only then invokes the model (detector). Mitigations could exist for adversarial attacks as they *typically* retain the sample's true class but rely on model feedback to exploit its imperfections, yielding "volatile" perturbations in its uncertainty regions that cause misclassification. This strategy has also been the focus of previous work on watermark removal [165, 120, 214]. Yet, if the watermark is actually removed, recovery would be impossible. As we claim that by simply disrupting the *carrier*, *UnMarker* removes the watermark, mitigations should fail.

To test this, we consider a *SOTA* mitigation— diffusion purification [182]. Purification inspires the *DiffusionAttack* (see §4.6.2), operating similarly: It adds noise, diluting adversarial perturbations, and uses a *DM* for reconstructing a pure (attack-free) sample. As this applies as a certifiable attack on *non-semantic* watermarks, these schemes cannot use this mitigation. Yet, these watermarks are of little importance since these proofs for their removal undermine their practicality. Since we aim to investigate the validity of defensive watermarking, we focus on the more reliable *semantic* schemes that meet the standards for utilizing purification as a defense. Still, *UnMarker*'s stages against both classes hinge on similar principles, making the findings applicable to defenses for *non-semantic* schemes if implemented.

We attack 100 samples for each scheme using *UnMarker* (*CHL*). Then, we perform purification and invoke the detector. Given the resistance of *semantic* schemes to purification (the *DiffusionAttack*) and the effectiveness of this mitigation against adversarial examples, if *UnMarker* does not remove the watermark but overshadows it with volatile changes as typical adversarial attacks behave, purification would eliminate *UnMarker* and restore the

watermark. Yet, the results verify *UnMarker*'s destructive nature that makes recovery impossible. For purification with t=0.1 (see §4.6.2), *StegaStamp*'s detection drops to 15% and *TRW*'s to 16%. A larger t leads to worse detection. Ironically, the defense degrades performance. While this is partly because diffusion may slightly decrease the *bit accuracy* (*inverse distance*), we apply it at half of its original strength (see §4.6.2), which allowed for ∼100% detection. The reason for this decline is *UnMarker*'s destruction, impairing the *DM*'s reconstruction ability. This raises the possibility of combining *UnMarker* with the *DiffusionAttack* as an even stronger attack, which we leave to future work due to the potential increased quality degradation.

Evaluating *AT* is prohibitive due to time limitations, as it requires numerous adversarial examples, and a single *UnMarker* attack takes several minutes. Yet, *AT* is only successful if the attack sample retains the watermark, which is not the case for *UnMarker*, as proven. As purification adheres to similar principles, its ineffectiveness implies the same for *AT*.

4.7.3 Outlook

A natural question is whether watermarking can ever be made reliable. Particularly, can we devise schemes that do not conform to our spectral analysis s.t. *UnMarker*, or similar attacks, would fail? The answer is likely no.

In §4.4, we establish that a *universal carrier* that encodes the watermark is necessary for any scheme due to computational considerations. Given the nature of visual content, the assertion that this *carrier* must be in the spatial structure or spectrally constructed is also invariant. We further show in §4.4.1 that spatial dependencies lead to unreliable watermarking behavior. Consequently, robust schemes must embed watermark signals in the *collective* spectral amplitudes. *Non-semantic* (invisible) watermarks preserve the content and predominantly embed their signal in high-frequency components. Yet, their footprints are limited in magnitude and fundamentally fragile: despite the slightly demanding nature of regeneration attacks, such methods can *certifiably* erase these watermarks, a limitation further exposed by direct spectral attacks such as *UnMarker*'s, which even require no pretrained generative models.

This restricts viable designs to content-influencing schemes (see §4.4.1), which modify the generated content. While such schemes may, in principle, embed watermark signals across both high- and low-frequency components, high-frequency signals inherit the limitations of *non-semantic* methods: Even if content modification enables increasing their magnitude, the inherently low energy concentration in these bands limits how much they can be amplified, leaving them fragile and likely susceptible to direct disruptions in this range of the spectrum,

such as *UnMarker*'s high-frequency stage (see §4.5.1), which removes such footprints using only small perturbations.

In contrast, *semantic* content-influencing schemes primarily embed watermark signals in lower-frequency components, entangling them with the underlying content structure to enhance robustness. Yet, they must simultaneously avoid introducing perceptible artifacts to retain quality and since obvious traces are removable even via naive editing. Existing *SOTA semantic* approaches already operate at this boundary, either introducing somewhat visible cues (*StegaStamp*) or significantly altering the content (*TRW*) to meet these constraints. Since *UnMarker* successfully removes such watermarks, future variants will have to exert increased control over the content, potentially limiting the space of watermarked images and reducing utility, or even degrading quality further, likely leaving them impractical. Combined with the limited effectiveness of mitigations, these findings call into question the practical viability of defensive watermarking and suggest that alternative approaches should be explored.

Toward a Resolution: While this evidence weighs against the viability of defensive watermarking, the question is not yet conclusively settled—and our analysis points to how it could be investigated in a principled manner, rather than left to speculation or an open-ended accumulation of attacks. Based on the above discussion, the central question can be re-framed as a structural one: *Can the entanglement between the watermark's spectral footprint and the underlying content be pushed far enough that sufficiently disrupting the carrier to eliminate it necessarily destroys that content?* The significance of our work lies in the methodology it offers for exploring this—specifically, for hardening the *carrier* itself.

The occasional success of common manipulations or prior advanced removal attacks stems from their *indirectly* affecting the *carrier*, weakening the embedded signature enough that it escapes detection. *UnMarker*, by contrast, identifies this channel and makes it the *immediate* optimization target. This distinction is what enables a constructive use: a scheme can incorporate robustness to *UnMarker*—or similar spectrally destructive attacks—as an explicit adversarial-training objective, pushing the watermark embedding toward its theoretical limit within the channel that actually carries the signal.

Two properties make robustness under such hardening meaningful rather than illusory. First, *UnMarker*'s vast hyperparameter space allows evaluating against a wide range of its variants, including configurations unseen during training, ruling out the possibility that apparent robustness merely reflects overfitting to a single attack instantiation. Second, because *UnMarker* targets the *carrier* directly and at its limit, robustness to it is a *necessary* condition for resisting removal by other techniques, which can also be validated under various destructive settings; the converse does not hold. Training against attacks that only

indirectly impact the *carrier* therefore risks a false sense of security, potentially protecting against a narrow class of threats alone.

Strong robustness measured after the direct-carrier hardening *UnMarker* facilitates would thus constitute defensible evidence for watermarking's viability. More importantly, if, even after such robustness-oriented training, evaluations reveal the watermark remains removable without meaningfully degrading quality, the fundamental untenability of defensive watermarking becomes extremely difficult to refute. In this sense, *UnMarker* is not merely a means of removing watermarks but a tool for resolving the feasibility question itself.

4.7.4 Limitations & Future Work

Limitations. First, we focus on practical (current) systems applicable to various images. Future schemes might be tailored for specific content (e.g., adding brand-specific glasses to face images), but such *GenAI* services will lack diversity. To meet users' needs, providers will also offer less restrictive tools, and attackers can use them instead, rendering content-specific watermarking useless. To reduce visibility and resist naive attacks, they must also employ spectral manipulations (for reasons similar to §4.4.1's), leaving *UnMarker* effective. Yet, for instance, without requiring universal applicability, the spectral footprint may target certain regions or objects only, not the collective spectrum. This necessitates augmenting *UnMarker* with content-based knowledge to identify optimal disruption regions, which could be studied should such schemes become prevalent. Second, *UnMarker* relies on accurate visual losses that preserve essential similarity cues while maximizing perturbations to attain the best removal rates. Future work may explore alternatives superior to ours (see §4.5.1). Third, while we consider visual content, our analysis translates to other domains (e.g., audio). Yet, *UnMarker* requires domain-based adaptations that can be interesting to investigate.

Forensics as a Countermeasure? Our analysis of mitigations in §4.7.2 was confined to recovery-based methods, consistent with watermarking's goal of remaining usable after an attack. A categorically different response—one that abandons recovery altogether—has recently been proposed [90, 76]: rather than restoring a removed watermark, a trained classifier attempts to detect the *footprints* left by a removal attack like *UnMarker*, flagging the image as suspicious even though the watermark itself is not restored. The crucial observation is that this does not revive watermarking as a defense—it replaces it. Detecting removal artifacts is a form of passive detection, returning to precisely the paradigm whose unreliability and poor generalization motivated watermarking in the first place (see §2.2.2).

As such, it inherits that paradigm's core weaknesses: limited generalization and susceptibility to specialized manipulations and adaptive attacks. Indeed, although Goonatilake and Ateniese [90] employ *UnMarker* as a primary case study to demonstrate this countermeasure's efficacy against its removal methodology, *UnMarker* was never optimized for stealth—i.e., to evade such forensic detectors. The pertinent risk to examine is therefore whether an *UnMarker* variant incorporating a stealth term in its objective would evade detection while still removing the watermark. Notably, these works [90, 76] themselves acknowledge that an attack jointly optimizing for removal and forensic stealth would likely be successful in doing so; combined with the documented fragility of passive detection, this renders the durability of such forensic defenses highly questionable. A systematic exploration of this direction is outside the scope of this thesis.

Watermark Forgery. Finally, a natural question is whether *UnMarker* extends to *forgery* attacks—embedding watermarks into arbitrary content. This threat is significant: prevalent forgeries erode trust in watermarking altogether (since genuine watermarks then certify nothing) and enable false attribution to the entity whose mark is forged [178]. As such, these attacks are currently an important active research area [70, 178, 15]. By construction, *UnMarker* removes rather than embeds signals. Still, it offers leverage for committing forgery. As demonstrated by An et al. [15], it is possible to train a surrogate classifier, and later adversarially optimize non-watermarked images against it to forge the watermark, with these attacks transferring to fool the actual watermark detector. However, this approach requires the surrogate to cleanly isolate the watermark channel, which in turn demands watermarked and non-watermarked samples from the *same prior* for training it, with the watermark's footprint being their only systematic difference. As explained in §4.2, assuming access to non-watermarked samples from that prior is not realistic. That said, *UnMarker* can approximate it in a practical manner—applied to watermarked images, it yields non-watermarked counterparts of the *same* content, differing only in the watermark and *UnMarker*'s own footprint. Pooling these with non-watermarked images from other distributions (e.g., publicly-available) and their "*UnMarked*" versions in the negative class as well can, in principle, suppress the bulk of residual *UnMarker*-specific cues, leaving the watermark as the distinguishing feature. This is an approximation rather than a guarantee—ground-truth same-prior negatives are unavailable, and it is possible that *UnMarker* may act differently on watermarked versus clean inputs, potentially reintroducing the confound the pooling aims to remove. We leave resolving this to future work.

4.7.5 Ethical Considerations

Misusing *GenAI* has severe impacts on society. We aim not to help generate harmful content but raise awareness regarding these systems' risks. Although we are not aware of providers that already released stable watermarking schemes, their adoption pace and attention from dominant entities (see §4.1) make assessments mandatory. We requested access to providers' *API*s for evaluation but never heard back. Thus, all studied systems are open-source.

4.8 Conclusion

We presented a novel attack against defensive watermarking via adversarial optimizations that disrupt the spectra of watermarked images and interfere with the watermarks' footprints, erasing them without interacting with the scheme. *UnMarker* is effective against all systems, including *semantic* schemes that have been assumed robust to such *query-free* and *black-box* attacks. Our results show that current defensive watermarking is not a viable countermeasure against deepfakes and that more robust defenses are needed.

Chapter 5

DiffBreak: Is Diffusion-Based Purification Robust?

This chapter is adapted from work published at the Conference on Neural Information Processing Systems (NeurIPS) 2025 [131]. It examines diffusion-based purification (*DBP*), a prominent defense paradigm against *AE*s, through the lens of *assumption stress-testing*, focusing on a crucial assumption: that systems grounded in strong theoretical arguments immediately inherit practical robustness. *DBP* has been regarded as a promising defense because it purifies *AE*s via diffusion models (*DM*s), motivated by their ability to project them back onto the natural manifold, eliminating malicious perturbations. While sound under idealized conditions, we show this assumption breaks down due to a fundamental gap between theory and practice, since instantiating *DM*s requires learned neural networks, which are themselves vulnerable to adversarial manipulation. Through a combination of theoretical analysis, corrected differentiation procedures, and improved evaluation protocols, we demonstrate that previous robustness claims were substantially inflated. We also propose an alternative statistically sound deployment protocol for *DBP* to mitigate variance-induced misclassifications, showing partial but meaningful robustness gain. Yet, we then present a novel low-frequency optimization strategy to craft systemic perturbations that survive *DBP*'s stochasticity, which collapses robustness to $\sim 0\%$ even in this stricter setup.

5.1 Motivation

ML classifiers are vulnerable to *adversarial examples (AEs)*—imperceptible perturbations that induce misclassification [233, 87]. *Adversarial Training* [167, 287] is attack-specific and

costly [265], while other defenses [96, 65, 26, 270, 273, 194, 278] are vulnerable to adaptive attacks [243]. *Diffusion-Based Purification (DBP)* [182, 250], which leverages *diffusion models (DMs)* [230, 227], has emerged as a promising defense [38, 267, 39, 286, 192]. *DBP* purifies inputs by solving the reverse stochastic differential equation (*SDE*) associated with *DM*s, diffusing the input with random noise to dilute adversarial changes and iteratively denoising it. Given *DMs'* data modeling abilities [63, 246], *DBP* operates under the assumption that each denoising (reverse) step's output belongs to a corresponding *marginal natural distribution* (a distribution obtained by introducing Gaussian noise into natural inputs, with decreasing variance proportional to that step).

This is the foundation for *DBP*'s *robustness* as it ensures its final outputs are from the natural distribution, while *AE*s lie outside this manifold and are unlikely to be preserved. Recent works treat *DBP* as a static pre-processor that standard gradient-based attacks cannot directly manipulate but rather only attempt to evade. Thus, attackers are assumed to either fail due to *DBP*'s randomness that masks the classifier's vulnerabilities [162], or be forced to rely on surrogate losses and heuristic approximations [124, 251] to *incrementally* boost performance. We theoretically refute *DBP*'s robustness rooted in its ability to perform projection onto the natural manifold in the presence of adaptive attacks: This is paradoxical as it assumes correct behavior of the score model $\boldsymbol{s}_\theta$ used by the *DM* (see §2.1.2) to mimic the gradients of the marginal distributions. Yet, $\boldsymbol{s}_\theta$ is an *ML* system with exploitable imperfections. In §5.3.1, we prove that **standard adaptive attacks** that simply backpropagate the classifier's loss gradients through *DBP* **effectively target $\boldsymbol{s}_\theta$ rather than the classifier**, forcing it to generate samples from an adversarial distribution. Hence, *DBP*'s theoretical motivation no longer holds.

Accordingly, in §5.3.2 and §5.3.3, we revisit *DBP*'s previously reported robustness, attributing it to backpropagation issues and improper evaluation protocols. Most prior works [124, 182, 162, 250] judge attacks based on a single purification of the final adversarial input. While this seemingly mirrors *DBP*'s intended "purify once and classify" deployment [182, 250], it is statistically invalid: since *DBP*'s purification process samples noise randomly, one evaluation of the *AE* fails to capture the true **single** misclassification probability across possible purifications. Moreover, due to *DBP*'s memory-intensive gradients, prior works implement exact backpropagation [124, 162, 251] via checkpointing [40]. We discover and fix issues in all previous implementations, introducing **DiffGrad**—the first reliable module for correct backpropagation through *DBP* (see Fig.5.1). In §5.4, we show that even under this one-shot *AE* evaluation protocol, *AutoAttack* [53] outperforms prior works, exposing their gradient fallacies. We further use a multi-sample (worst-case) evaluation that purifies multiple copies of the generated *AE* and declares success if any of them is misclassified, capturing *DBP*'s stochasticity and resubmission risk properly—aligned

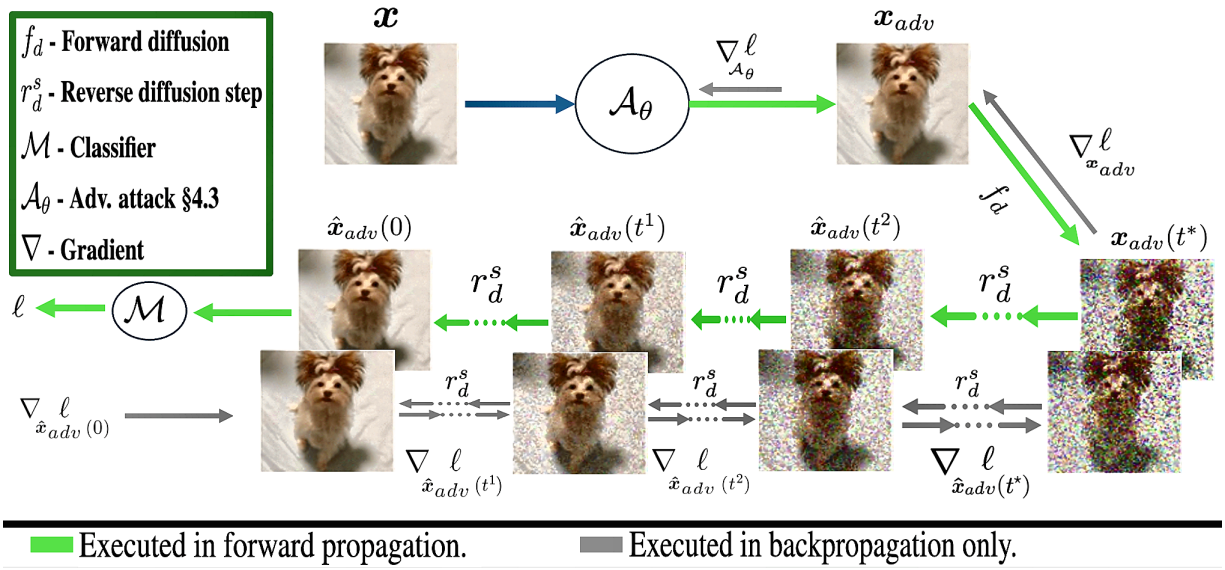


Figure 5.1: **DiffGrad**. $\boldsymbol{x}$ is given to the iterative attack algorithm $\mathcal{A}_\theta$ to generate $\boldsymbol{x}_{adv}$. At each iteration, $\boldsymbol{x}_{adv}$ is propagated through *DBP*, yielding $\hat{\boldsymbol{x}}_{adv}(0)$ that is given to $\mathcal{M}$ while storing each intermediate $\hat{\boldsymbol{x}}_{adv}(t)$ (bottom replicas) from *DBP*'s reverse pass (see §5.2) but without saving any graph dependencies. Backpropagation uses the stored samples: Starting from $t=-dt$ ($dt<0$—see §5.2), each $\hat{\boldsymbol{x}}_{adv}(t)$ is used to recompute $\hat{\boldsymbol{x}}_{adv}(t+dt)$, retrieving the required dependencies. Then, we recursively obtain the gradient w.r.t. $\hat{\boldsymbol{x}}_{adv}(t)$ from the gradients w.r.t. $\hat{\boldsymbol{x}}_{adv}(t+dt)$ using this recovered sub-graph (see §5.3.2). Finally, gradients are backpropagated in a standard manner from $\boldsymbol{x}_{adv}(t^*)$ to $\mathcal{A}_\theta$ to update $\boldsymbol{x}_{adv}$.

with Wang et al. [251], who also identified this evaluation gap, but retained problematic gradients. By fixing both issues, we prove *DBP* even more vulnerable than Wang et al. [251] report, with robustness dropping below 17.19%.

Additionally, we observe that *DBP*'s vulnerability to the resubmission risk is inherent to the current variance-fragile deployment scheme, wherein the defender is assumed to only purify and then classify a single copy of the incoming sample. To replace this setup, we propose a more robust majority-vote (*MV*) alternative, where the defender purifies multiple copies through different random paths and then classifies them, making the final decision based on the majority label. This indeed boosts *DBP*'s resilience to *AE*s. Yet, even under *MV*, *DBP* retains only **partial** robustness against traditional norm-bounded attacks with our correct backpropagation—see §5.4.2. Notably, we also show that prior proposals of **advanced** adaptive attack enhancements tailored to *DBP* (e.g., **DiffHammer** [251], **DiffAttack** [124]), being oblivious to the ability we identify of standard gradient-based

attacks to reshape its behavior, in fact, disrupt optimization and lower success rates. This degraded performance of *DBP* even under *MV* further confirms our theoretical scrutiny.

Finally, as we established the invalidity of the "intermediate projection" hypothesis, we claim that the residual (partial) resistance of *DBP* under *MV* is merely due to *DBP*'s vast stochasticity: common (traditional) *AE*s that introduce large changes to *individual* pixels are diluted and may not impact most random paths. Instead, we require stealthy modifications that affect many pixels simultaneously. In §5.5, we propose a novel adaptation of our recent adversarial optimization strategy, namely *UnMarker* [130] (see Chapter 4), from the image watermark removal domain to instantiate the standard adaptive attack, which crafts low-frequency perturbations, accounting for links between neighboring pixels. This technique completely *DBP* even under *MV*.

We make the following contributions:

- We analytically scrutinize standard adaptive attacks on *DBP*, proving they nullify its theoretical robustness motivation.
- We address protocol and gradient issues in prior work, enabling reliable evaluations, and demonstrating degraded performance of *DBP* and the ineffectiveness of recent attack enhancement strategies [124, 251].
- We introduce and evaluate a statistically grounded *MV* deployment protocol for *DBP*.
- We propose and adapt low-frequency (*LF*) optimizations to *DBP*, achieving unprecedented success even under *MV*.
- **Availability**: Aside from scalability and backpropagation issues, existing *DBP* implementations and attacks lack generalizability. We provide *DiffBreak*[1]—the first toolkit for evaluating any classifier with *DBP* under various optimization methods, including our novel *LF*, using our reliable **DiffGrad** module for backpropagation.
- We perform extensive evaluations on ***ImageNet*** and ***CIFAR-10***, proving our methods **defeat the *DBP* defense paradigm across several variants**, bringing robustness to $\sim 0\%$, and outperforming previous work by a large margin.

[1] https://github.com/andrekassis/DiffBreak

5.2 Background & Related Work

Diffusion Models (*DM*s) [227, 230]. We briefly summarize the aspects of *DM*s relevant to this chapter and refer the reader to §2.1.2 for a comprehensive background. *DM*s are state-of-the-art generative models consisting of two stochastic processes: a **forward diffusion (pass)**, which over time perturbs a clean sample $\boldsymbol{x}$ from a known distribution p gradually into noise and a **reverse denoising process (pass)** that iteratively reconstructs samples in p from noisy inputs. The forward pass (i.e., noise introduction) is straightforward and can even be replaced with a single operation following the closed-form expression given in *eq.* (2.2). The core process is the reverse pass, which enables the use of *DM*s to generate natural data samples from pure noise but, while it can be modeled mathematically as a stochastic differential equation (*SDE*), it lacks a closed-form integral representation and must be performed iteratively over small negative timesteps $dt < 0$, evolving over noisy intermediate states $\hat{\boldsymbol{x}}(t)$ toward a clean reconstruction $\hat{\boldsymbol{x}}(0)$. Each denoising step invokes a learned score model $\boldsymbol{s}_\theta$ to approximate the gradient $\nabla_{\hat{\boldsymbol{x}}(t)} \log p_t(\hat{\boldsymbol{x}}(t))$ of the marginal density and guide the transition between consecutive states.

Diffusion-Based Purification (*DBP*) [182, 272, 250, 280] utilizes diffusion models (*DM*s) to purify adversarial examples. Instead of using *DM*s in the originally intended manner to generate fresh data samples from pure random noise, *DBP* diffuses the incoming input $\boldsymbol{x}$ until optimal time t^* (i.e., partially applies the forward pass using *eq.* (2.2)) that preserves class semantics while still diminishing malicious interruptions. The noisy $\boldsymbol{x}(t^*)$ is then given to the reverse process as its input $\hat{\boldsymbol{x}}(t^*) := \boldsymbol{x}(t^*)$, which iteratively reconstructs a clean $\hat{\boldsymbol{x}}(0) \approx \boldsymbol{x}$ expected to lie on the natural data manifold, filtering out—with high probability—adversarial perturbations that are distributional outliers, and enabling correct classification of the resulting $\hat{\boldsymbol{x}}(0)$—see §5.3.1.

Certified vs. Empirical Robustness. While recent work has explored certified guarantees for *DBP* via randomized smoothing (RS) [272, 35, 288, 38, 267], these guarantees hold only under restrictive assumptions—e.g., small ℓ_2 perturbations and thousands of Monte Carlo samples per input. In contrast, empirical defenses [250, 182, 192, 39, 286] aim for practical robustness under more realistic threats [53] and efficient inference. Like prior work [251, 124, 182, 250, 148], we explicitly target these empirical defenses—not orthogonal certified variants whose guarantees do not hold under stronger practical perturbations or operational constraints. Certification protocols are computationally prohibitive, and their bounds fail to capture threats that remain imperceptible but exceed the certified radii.

Adversarial Attacks under Preprocessing. *DBP* as an empirical defense operates as a preprocessing layer applied prior to classification. Here, the effective system the attacker encounters becomes the composition of the defense and the classifier and we can extend **Definition** 2.3.1 to formalize adversarial attacks as follows:

Definition 5.2.1 (Adversarial Attacks under Preprocessing)**.** *Given $\boldsymbol{x} \in \mathbb{R}^d$ with true label y, classifier $\mathcal{M}$, and preprocessing defense G, attackers aim to generate a similar $\boldsymbol{x}_{adv}$ s.t. $\mathcal{M}(G(\boldsymbol{x}_{adv})) \neq y$. Formally, the adversarial optimization problem is given as:*

$$\boldsymbol{x}_{adv} = \underset{\boldsymbol{\mathcal{D}}(\boldsymbol{x}', \boldsymbol{x}) \leqslant \epsilon_{\boldsymbol{\mathcal{D}}}}{\arg\min} \; \mathbb{E}[\ell_G^{\mathcal{M}}(\boldsymbol{x}',\ y)]$$

*for $\ell_G^{\mathcal{M}}(\boldsymbol{x}, y) = \ell(\mathcal{M}(G(\boldsymbol{x})), y)$, where ℓ is a loss over $\mathcal{M}$'s output that captures the desired outcome as in **Definition** 2.3.1 and $\boldsymbol{\mathcal{D}}$ is a distance metric that ensures similarity if kept below some $\epsilon_{\boldsymbol{\mathcal{D}}}$. The expected value accounts for potential stochasticity in G (e.g., DBP).*

As explained in §2.3.1, this formulation captures *untargeted* attacks. In this chapter, when studying *DBP*, we focus on these attacks following previous work [182, 250, 194, 171].

For such preprocessing mechanisms, proper evaluation requires adaptive attacks—see §2.3.2. That is, strong attacks must account for the defense and attempt to bypass it. Such adaptive attacks generally cannot be uniquely defined since they are inherently tailored to the specific defense and may involve specialized objectives or techniques.

Nevertheless, a fundamental principle for evaluating preprocessing defenses is to at least incorporate the defense into the standard (gradient-based) attack pipelines that are known to be effective against undefended classifiers, and update inputs based on feedback from the composed system (defense and classifier). In comparison, attacks crafted solely against the classifier and evaluated post hoc on the defended system have often led to inflated robustness estimations [34]. We refer to this minimal yet essential form of adaptation as the **standard adaptive attack**. Below, we extend **Definition** 2.3.3 to formalize this setting in the presence of preprocessing.

Definition 5.2.2 (Standard Adaptive Gradient-Based Attack)**.** *Let $\mathcal{M}$ be a classifier, G a preprocessing defense as in **Definition** 5.2.1, and $\ell(\mathcal{M}(\boldsymbol{x}), y)$ be the same classifier-output loss used by a standard gradient-based attack against the undefended classifier as in **Definition** 2.3.3. The corresponding standard adaptive gradient-based attack against the defended pipeline is obtained by composing the defense with the classifier and directly propagating gradients through the resulting system:*

$$\mathbb{E}[\nabla_{\boldsymbol{x}} \ell(\mathcal{M}(G(\boldsymbol{x})), y)]$$

The expected value captures potential stochasticity in G, as in ***Definition*** *5.2.1. The similarity constraint $\boldsymbol{\mathcal{D}}$ may be enforced through different optimization mechanisms depending on the attack, including explicit projection steps, clipping operations, constrained parameterizations, or additional regularization terms incorporated into the optimization objective.*

Remark (Scope of the Standard Adaptive Attack). This formulation does *not* introduce defense-specific auxiliary objectives, intermediate-state optimization targets, defense-specific surrogate losses, or modified optimization criteria. Adaptation is achieved solely by inserting the defense into the differentiable attack pipeline and computing gradients through the composed system. This boundary is what distinguishes the standard adaptive attack from *advanced adaptive attacks*, which modify the attack structure itself through additional heuristics, objectives, or altered formulations and procedures (see §5.4).

5.3 Revisiting Diffusion-Based Purification

We study standard gradient-based attacks. In §5.3.1, we theoretically prove such adaptive attacks invalidate *DBP*'s principles. Then, we reconcile this with previous findings of *DBP*'s robustness, attributing them to backpropagation issues and improper evaluation protocols. In §5.3.2, we analyze backpropagation mismatches in previous work and propose fixes. Finally, we present an improved evaluation protocol for *DBP* in §5.3.3.

5.3.1 Why *DBP* Fails: Theoretical Vulnerability

DBP's robustness is primarily attributed to its ability to project the purified $\hat{\boldsymbol{x}}(0)$ onto the natural manifold [182, 272, 124]. This foundational assumption—rooted in the inherent behavior of the purification process—has remained unchallenged. In fact, the recent advanced adaptive attacks have been explicitly built around it: **DiffAttack** [124] concedes this property and thus attempts to resist it through external, yet suboptimal, losses, while **DiffHammer** [251] constrains its optimization only to feasible paths that are more likely to withstand this assumed projection (see §5.4). This assumption is often justified through *DBP*'s marginal consistency: since $\boldsymbol{s}_\theta \approx \nabla \log p_t$ (see §2.1.2), the intermediate noisy sequences $\{\boldsymbol{x}(t)\}_{t\in[0,1]}$ and $\{\hat{\boldsymbol{x}}(t)\}_{t\in[0,1]}$ realized during both the forward and the reverse diffusion passes follow the same marginals [17], yielding $\hat{\boldsymbol{x}}(0) \sim p$. Specifically, Xiao et al. [272] show that:

$$\Pr(\hat{\boldsymbol{x}}(0)|\boldsymbol{x}) \propto p(\hat{\boldsymbol{x}}(0)) \cdot e^{-\frac{\alpha(t^*)\|\hat{\boldsymbol{x}}(0)-\boldsymbol{x}\|_2^2}{2(1-\alpha(t^*))}}$$

where p is the density of the natural data distribution and $\alpha(t^*)$ is the variance schedule at time t^*—see *eq.* (2.2). Thus, *DBP* is expected to reject adversarial examples by construction: the probability of producing any $\hat{\boldsymbol{x}}(0)$ that is adversarial is extremely low, as it is proportional to its density under the natural distribution, which is small since such samples lie off-manifold.

However, while this holds under ideal functionality of the score model $\boldsymbol{s}_\theta$, in practice, $\boldsymbol{s}_\theta$ is itself an *ML* model—differentiable and susceptible to adversarial manipulation. Our key insight is that standard gradient-based (adaptive) attacks, when backpropagated through *DBP*, do not simply strive to bypass the purifier as previously thought: under that prevailing view, the effective target of these attacks is the classifier, but backpropagating through *DBP* aims to learn perturbations that slip through purification by exploiting blind spots of this $\boldsymbol{s}_\theta$. In that case, *DBP* would indeed drastically limit the attack's efficacy due to its many repeated denoising steps, making evading the entire invocation sequence simultaneously difficult. This reasoning aligns with prior work reporting high robustness and upholding the theoretical motivation. In contrast, we show that these attacks, in fact, accomplish something far more significant—they implicitly turn the score model $\boldsymbol{s}_\theta$ itself into the primary target, controlling it to generate samples from an adversarial distribution. This challenges the assumptions underlying not only *DBP*'s robustness, but also prior advanced attack strategies, many of which [251, 124] distort gradients or constrain optimization in ways that ignore the score model's vulnerability, degrading their own effectiveness. Below, we formally characterize this vulnerability.

Definitions. *Diffusion Process & Score Model.* For $t_1 \geqslant t_2$, let $\hat{\boldsymbol{x}}_{t_1:t_2}$ represent the joint reverse diffusion trajectory $\hat{\boldsymbol{x}}(t_1), \hat{\boldsymbol{x}}(t_1 + dt), \ldots, \hat{\boldsymbol{x}}(t_2)$. Let $\boldsymbol{s}_\theta$ denote the score model used by *DBP* to approximate the gradients of the natural data distribution at different time steps. $\boldsymbol{s}_{\theta^t}$ is the abstract score model invoked at time step t with parameters θ^t, corresponding to $\boldsymbol{s}_\theta(\cdot, t)$. Given that the score model $\boldsymbol{s}_\theta$ is an *ML* model that interacts with adversarial input $\boldsymbol{x}$, we further denote the parameters at each reverse step as $\theta^t_{\boldsymbol{x}}$, capturing their dependence on $\boldsymbol{x}$. That is, $\theta^t_{\boldsymbol{x}} \equiv \theta^t(\boldsymbol{x})$, which makes explicit that the purification process is **not immutable**, as adversarial modifications to $\boldsymbol{x}$ can shape its behavior.

Classifier. For a classifier $\mathcal{M}$ and label y, let $\mathcal{M}^y(\boldsymbol{u})$ denote the probability that $\boldsymbol{u}$ belongs to class y.

Standard Adaptive Attack Step. A gradient-based update step that iteratively modifies $\boldsymbol{x}$ following **Definition** 5.2.2. Each update is performed with a small learning rate $\eta > 0$:

$$\boldsymbol{x} = \boldsymbol{x} - \eta \widetilde{\nabla}_{\boldsymbol{x}}, \quad \text{where} \quad \widetilde{\nabla}_{\boldsymbol{x}} = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{x}} [\mathcal{M}^y(\hat{\boldsymbol{x}}(0)_n)].$$

Here, $\widetilde{\nabla}_{\boldsymbol{x}}$ is the empirical expected gradient estimate over N purified samples $\hat{\boldsymbol{x}}(0)_n$, obtained via Monte Carlo approximation and backpropagated through *DBP*'s stochastic process. For clarity, we state the result for the unconstrained update rule above, as the core argument concerns the gradient flow through *DBP*'s reverse process rather than the specific mechanism used to enforce similarity constraints. In practice, standard adaptive attacks typically incorporate projection, clipping, constrained parameterizations, or regularization terms to satisfy the perturbation budget in **Definition** 5.2.2; these additions do not alter the theoretical argument and the result extends naturally to such constrained settings.

Theorem 5.3.1. *The **standard** adaptive attack **optimizes the entire reverse diffusion process**, modifying the parameters $\{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t^*}$ such that the output distribution $\hat{\boldsymbol{x}}(0) \sim DBP^{\{\theta^t_{\boldsymbol{x}}\}}(\boldsymbol{x})$, where $DBP^{\{\theta^t_{\boldsymbol{x}}\}}(\boldsymbol{x})$ is the DBP pipeline with the score model's weights $\{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t^*}$, is adversarially aligned. That is, the adversary **implicitly controls the purification path** and optimizes the weights $\{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t^*}$ to maximize:*

$$\max_{\{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t^*}} \mathbb{E}_{\hat{\boldsymbol{x}}(0)\sim DBP^{\{\theta^t_{\boldsymbol{x}}\}}(\boldsymbol{x})}[\Pr(\neg y|\hat{\boldsymbol{x}}(0))].$$

*Since $\{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t^*}$ depend on the purification trajectory $\hat{\boldsymbol{x}}_{t^*:0}$, optimizing $\boldsymbol{x}$ under the adaptive attack **directly shapes the purification process itself**, forcing DBP to generate adversarially structured samples that maximize misclassification.*

Impact. As *DBP* is widely perceived as immutable, its robustness is often attributed to gradient masking (vanishing or explosion) due to its presumed ability to project inputs onto the natural manifold. However, this reasoning treats *DBP* as a **static pre-processing step** rather than a **dynamic optimization target**. Our result **overturns this assumption**: rather than resisting adversarial influence, *DBP* itself becomes an **active participant in adversarial generation**. Crucially, standard adaptive attacks that backpropagate accurate gradients through *DBP* **do not merely attempt to evade it through perturbations that strive to survive purification as is commonly believed** [124, 251, 272, 162]—they implicitly **exploit and reshape** its behavior, forcing the purification process to align with the attacker's objective. The introduced perturbations **directly shape the purification trajectory**, causing *DBP* to generate, rather than suppress, adversarial samples. Thus, *DBP* does not inherently neutralize adversarial perturbations but shifts the optimization target from the classifier to the score model $\boldsymbol{s}_\theta$, leaving this paradigm vulnerable.

Proof. The adversary aims to maximize:

$$\mathcal{Q}(\boldsymbol{x}) \equiv \mathbb{E}_{\hat{\boldsymbol{x}}(0)\sim DBP^{\{\theta^t\}}(\boldsymbol{x})}[\Pr(\neg y|\hat{\boldsymbol{x}}(0))]$$

Expanding this expectation yields the following alternative representations:

$$\begin{aligned}\mathcal{Q}(\boldsymbol{x}) &= \int \Pr(\neg y|\hat{\boldsymbol{x}}(0))p(\hat{\boldsymbol{x}}(0))d\hat{\boldsymbol{x}}(0) \\ &= \int \Pr(\neg y|\hat{\boldsymbol{x}}(0)) \int p(\hat{\boldsymbol{x}}(0)|\hat{\boldsymbol{x}}_{t*:-dt})p(\hat{\boldsymbol{x}}_{t*:-dt})d\hat{\boldsymbol{x}}_{t*:-dt}d\hat{\boldsymbol{x}}(0) \\ &= \int \Pr(\neg y|\hat{\boldsymbol{x}}(0))p(\hat{\boldsymbol{x}}_{t*:0})d\hat{\boldsymbol{x}}_{t*:0}.\end{aligned}$$

The first transition is due to the definition of expectation, the second follows from the definition of marginal probability, and the final transition replaces the joint density function with its compact form.

Since this formulation abstracts away $\boldsymbol{x}$ and $\{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t*}$, we explicitly include them as $p(\hat{\boldsymbol{x}}_{t*:0}) \equiv p(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x}, \{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t*})$. For simplicity, we define $p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x}) \equiv p(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x},\ \{\theta^t_{\boldsymbol{x}}\}_{t\leqslant t*})$. Thus, the final optimization objective is:

$$\mathcal{Q}(\boldsymbol{x}) = \int \Pr(\neg y|\hat{\boldsymbol{x}}(0))p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x})d\hat{\boldsymbol{x}}_{t*:0}.$$

Since we assume smoothness ($\mathcal{C}^2$), we interchange gradient and integral:

$$\nabla_{\boldsymbol{x}}[\mathcal{Q}(\boldsymbol{x})] = \int \Pr(\neg y|\hat{\boldsymbol{x}}(0))\nabla_{\boldsymbol{x}}p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x})d\hat{\boldsymbol{x}}_{t*:0}.$$

where the last step is because $\Pr(\neg y|\hat{\boldsymbol{x}}(0))$ does not depend on $\boldsymbol{x}$, but only on $\hat{\boldsymbol{x}}(0)$ which is independent of $\boldsymbol{x}$ (though its probability of being the final output of *DBP* is a function of $\boldsymbol{x}$). Optimizing the objective via gradient ascent relies on altering *DBP*'s output distribution alone, evident in its gradient where $\Pr(\neg y|\hat{\boldsymbol{x}}(0))$ assigned by the classifier is a constant with its gradient ignored. While we lack direct access to the gradients of the probabilistic paths above, we may still attempt to solve the optimization problem.

Recall that by definition, $\forall \boldsymbol{u},\ \mathcal{M}^y(\boldsymbol{u}) = \Pr(y|\boldsymbol{u})$. Thus, the required gradient can also be expressed as:

$$\nabla_{\boldsymbol{x}}\big[\mathcal{Q}(\boldsymbol{x})\big] = \nabla_{\boldsymbol{x}}\Bigg[\int \Pr(\neg y|\hat{\boldsymbol{x}}(0))p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x})d\hat{\boldsymbol{x}}_{t*:0}\Bigg] =$$

$$\nabla_{\boldsymbol{x}}\Bigg[\mathop{\mathbb{E}}_{p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x})}\left[1-\mathcal{M}^y(\hat{\boldsymbol{x}}(0))\right]\Bigg] = -\nabla_{\boldsymbol{x}}\Bigg[\mathop{\mathbb{E}}_{p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t*:0}|\boldsymbol{x})}\left[\mathcal{M}^y(\hat{\boldsymbol{x}}(0))\right]\Bigg]$$

Yet, this expectation is over probabilistic paths whose randomness is due to the noise $\boldsymbol{\epsilon}$ of the forward pass and the Brownian motion $d\bar{\boldsymbol{w}}_t$ at each reverse step t that are independent.

Hence, by the law of the unconscious statistician:

$$-\nabla_{\boldsymbol{x}}\left[\mathop{\mathbb{E}}_{p_{\theta_{\boldsymbol{x}}}(\hat{\boldsymbol{x}}_{t^*:0}|\boldsymbol{x})}[\mathcal{M}^y(\hat{\boldsymbol{x}}(0))]\right] = -\nabla_{\boldsymbol{x}}\left[\mathop{\mathbb{E}}_{p_r(\boldsymbol{\epsilon},d\bar{\boldsymbol{w}}_{t^*},\ldots,d\bar{\boldsymbol{w}}_0)}[\mathcal{M}^y(\hat{\boldsymbol{x}}(0))]\right]$$

where $p_r(\boldsymbol{\epsilon}, d\bar{\boldsymbol{w}}_{t^*}, ..d\bar{\boldsymbol{w}}_0)$ denotes the joint distribution of the noise $\boldsymbol{\epsilon}$ and the Brownian motion vectors in the reverse pass. Note that $\hat{\boldsymbol{x}}(0)$ on the RHS denotes the output obtained by invoking the *DBP* pipeline with some assignment of these random vectors on the sample $\boldsymbol{x}$. As earlier, we can interchange the derivation and integration, obtaining:

$$-\mathop{\mathbb{E}}_{p_r(\boldsymbol{\epsilon},d\bar{\boldsymbol{w}}_{t^*},\ldots,d\bar{\boldsymbol{w}}_0)}\left[\nabla_{\boldsymbol{x}}[\mathcal{M}^y(\hat{\boldsymbol{x}}(0))]\right].$$

Now, let $\boldsymbol{G}^{\boldsymbol{x}}$ denote the random variable that is assigned values from $\nabla_{\boldsymbol{x}}\mathcal{M}^y(\hat{\boldsymbol{x}}(0))$, where $\hat{\boldsymbol{x}}(0)$ is as described above, and denote its covariance matrix by $\Sigma_{\boldsymbol{G}^{\boldsymbol{x}}}$. Essentially, we are interested in $\mathbb{E}[\boldsymbol{G}^{\boldsymbol{x}}]$. If we define $\widetilde{\nabla}_{\boldsymbol{x}}$ as:

$$\widetilde{\nabla}_{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N}\nabla_{\boldsymbol{x}}[\mathcal{M}^y(\hat{\boldsymbol{x}}(0)_n)]$$

where each $\hat{\boldsymbol{x}}(0)_n$ is the output of *DBP* invoked with a certain random path $(\boldsymbol{\epsilon}^n, d\bar{\boldsymbol{w}}^n_{t^*}, ..d\bar{\boldsymbol{w}}^n_0) \overset{i.i.d}{\sim} p(\boldsymbol{\epsilon}, d\bar{\boldsymbol{w}}_{t^*}, ..d\bar{\boldsymbol{w}}_0)$ and N is a sufficiently large number of samples. Then due to the central limit theorem, $\widetilde{\nabla}_{\boldsymbol{x}} \longrightarrow \mathcal{N}(\mathbb{E}[\boldsymbol{G}^{\boldsymbol{x}}], \frac{\Sigma_{\boldsymbol{G}^{\boldsymbol{x}}}}{N})$. That is, propagating multiple copies through *DBP* and then averaging the loss gradients computes the required gradient for forcing *DBP* to alter its output distribution. The larger the number N of samples is, the lower the variance of the error, as can be seen above. Note that the standard adaptive attack step operates exactly in this manner, assuming it uses $\mathcal{M}^y$ as the loss it minimizes (but the soundness of the approach generalizes intuitively to any other loss with the same objective over the classifier's logits), proving our claim. □

5.3.2 Precise *DBP* Gradients with DiffGrad

Building on our theoretical analysis, we examine why prior adaptive attacks [182, 250, 124, 162, 148] underperform and show how **DiffGrad**, provided by our *DiffBreak* toolkit, resolves these limitations.

We reserve the terms **forward/reverse pass (process)** for *DBP* (see §5.2) and **forward/back propagation** for gradient computations. Let $\hat{\boldsymbol{x}}(t^*) := \boldsymbol{x}(t^*)$. Following §2.1.2, the reverse process follows:

$$\hat{\boldsymbol{x}}(t+dt) = \hat{\boldsymbol{x}}(t) + d\hat{\boldsymbol{x}}(t), \quad dt < 0, \tag{5.1}$$

and by chain rule the gradient of any function F of $\hat{\boldsymbol{x}}(t+dt)$ w.r.t. $\hat{\boldsymbol{x}}(t)$ is:

$$\nabla_{\hat{\boldsymbol{x}}(t)} F = \nabla_{\hat{\boldsymbol{x}}(t)} \langle \hat{\boldsymbol{x}}(t+dt), \nabla_{\hat{\boldsymbol{x}}(t+dt)} F \rangle. \tag{5.2}$$

Applying this recursively yields gradients w.r.t. $\hat{\boldsymbol{x}}(t^*)$, and via *eq.* (2.2), the input $\boldsymbol{x}$. Yet, standard automatic differentiation is impractical for *DBP* due to excessive memory overhead from dependencies between all $\hat{\boldsymbol{x}}(t)$. Prior work resorts to approximations such as the **adjoint** method [155], or checkpointing [40] to compute the exact gradients. However, approximations yield suboptimal attack performance [148]. On the other hand, we identify critical issues in existing checkpointing-based implementations that previously led to inflated estimates of *DBP*'s robustness (see Appendix C.2 for detailed analysis, **DiffGrad**'s pseudo-code, and empirical evaluations of backpropagation issues):

1) High-Variance Gradients: Gradient-based attacks require estimating expected gradients using N Monte Carlo (EOT) samples. When N is small, gradient variance is high, leading to unreliable updates. Prior works used small N due to limitations in how EOT samples were purified—one at a time via serial loops in standard attack benchmarks (e.g., *AutoAttack* [53]). **DiffGrad** parallelizes EOT purifications. This integrates seamlessly with standard attacks. Coupled with our termination upon success protocols—see §5.4.2, this enables us to use up to 128 EOT samples (for ***CIFAR-10***) with a speedup of up to 41× upon early termination, drastically reducing variance.

2) Time Consistency: torchsde[2] is the de facto standard library for *SDE* solvers. Hence, several prior checkpointing approaches likely use *torchsde* as their backend. Yet, *torchsde* internally converts the integration interval into a *PyTorch* tensor, which causes a discrepancy in the time steps on which the score model is invoked during both propagation phases due to rounding issues if the checkpointing module is oblivious to this detail. **DiffGrad** ensures time steps match in both phases.

3) Reproducing Stochasticity: Forward and backward propagation should reuse identical randomness to preserve fidelity. In vanilla *DBP*, diffusion noise cancels in the gradient, so not preserving stochastic components has no measurable effects. Yet, for guided schemes whose computations depend on randomness (some current ones and potential future variants), failing to reproduce the noise realizations can bias gradients and derail optimization. We thus introduce a structured *Noise Sampler* $\boldsymbol{NS}$ that records all noise during the forward pass and reuses it in backpropagation. We also enable deterministic CuDNN kernel selection to avoid backend nondeterminism—see Appendix C.2.1 for details.

4) Guidance Gradients: Guided *DBP* [250] uses guidance metrics along the reverse pass. This guidance is obtained by applying a guidance function $\boldsymbol{g_{fn}}$ to (potentially) both the

[2] https://github.com/google-research/torchsde

original input $\boldsymbol{x}$ and the reverse-diffused $\hat{\boldsymbol{x}}(t)$. As such, it creates paths from $\boldsymbol{x}$ to the loss that basic checkpointing fails to consider. Furthermore, the guidance itself is often in the form of a gradient, necessitating second-order differentiation during backpropagation, which is, by default, disabled by differentiation engines. We extend **DiffGrad** to include these "guidance" gradients. Critically, recent SOTA *DBP* defenses generate purified outputs from pure noise, relying on $\boldsymbol{x}$ only via guidance (see §5.6). As such, the lack of support for guidance gradients in all previous implementations leads to a false sense of security. To our knowledge, **DiffGrad** is the first to account for this component.

5) Proper Use of The Surrogate Method [148]: Upon inspecting **DiffHammer**'s code, we find their checkpointing implementation effectively applies Lee and Kim [148]'s **surrogate** approximation to calculate the gradients, which already slightly degrades gradient quality with no practical performance gains—see Appendix C.4. Yet, the real issue lies in **DiffHammer**'s implementation of the **surrogate**, which diverges from [148], further corrupting the gradients. We defer details to Appendix C.2.1.

5.3.3 On The Issues in *DBP*'s Evaluation Protocols.

DBP is typically deployed in a single-purification (*SP*) setting: an input $\boldsymbol{x}$ is purified once via a stochastic reverse process and classified as $c = \mathcal{M}(\hat{\boldsymbol{x}}(0))$ [182, 250]. This assumes $\hat{\boldsymbol{x}}(0)$ lies on the natural manifold, yielding a correct label with high probability—see §5.3.1. Yet, prior studies [124, 182, 250] fail to assess this very property: they evaluate attacks by purifying only the final adversarial iterate, testing just a single stochastic path, yielding noisy, unrepresentative estimates.

DiffHammer [251] addressed this with worst-case robustness (Wor.Rob): the attack succeeds if any of N purifications (evaluated at each attack step) leads to misclassification. The metric is Wor.Rob $:= 1 - \frac{1}{S}\sum_{j=1}^{S} \underset{i\in[\![N]\!]}{max} \mathcal{A}_i^{(j)}$, where $\mathcal{A}_i^{(j)} = 1$ if the i-th purification of sample j fails, and S is the dataset size. They justify this protocol by modeling resubmission attacks (e.g., in login attempts), thereby limiting N to (typically) $N = 10$ after which the attacker will be blocked. However, attackers can retry inputs arbitrarily in stateless settings like spam, CSAM, and phishing. Even in stateful systems, the defender has no control over the stochastic path. Thus, *SP* evaluations require many purifications per input for statistically meaningful conclusions.

Yet, any stochastic defense fails given enough queries [163]. Even if *DBP* nearly always projects to the manifold, *SP* is only reliable if misclassification probability is negligible over all paths—rare in practice (§5.4). Robust predictions must hence aggregate over multiple purifications. We thus propose:

Majority Vote (MV). Given input $\boldsymbol{x}$, generate K purifications and predict by majority: $c = MV^K(\boldsymbol{x}) := mode\{\mathcal{M}(\hat{\boldsymbol{x}}^{(1)}(0)), \ldots, \mathcal{M}(\hat{\boldsymbol{x}}^{(K)}(0))\}$. Unlike certified smoothing [272], *MV* requires no excessive sampling, yet offers a stable, variance-tolerant robustness estimate, albeit without formal guarantees. We use the majority-robustness metric: MV.Rob $:= 1 - \frac{1}{S}\sum_{j=1}^{S} \mathbb{1}\left[MV^K(\boldsymbol{x}_j) \neq y_j\right]$, where y_j is the ground-truth label. **DiffHammer**'s additional Avg.Rob $:= 1 - \frac{1}{NS}\sum_{j=1}^{S}\sum_{i=1}^{N} \mathcal{A}_i^{(j)}$ averages path-level failures across all samples, conflating individual copy failures with sample robustness. A few fragile samples can dominate this metric, making it unreliable for deployment.

5.4 Experiments

We reevaluate *DBP*, demonstrating its degraded performance when evaluation and back-propagation issues are addressed, cementing our theoretical findings from §5.3.1.

Setup. We evaluate on ***CIFAR-10*** [139] and ***ImageNet*** [62] similar to previous work [251, 124, 182, 250]. We consider two foundational *DBP* defenses: The **VP-SDE** *DBP* (*DiffPure*) [182] and the *Guided*-**DDPM** (see §5.3.2), *GDMP* [250]. We use the *DM*s [63, 106, 230] studied in the original works, adopting the same purification settings—See Appendix C.3. Following Nie et al. [182], we use WideResNet-28-10 and WideResNet-70-16 [284] classifiers for ***CIFAR-10***, and WideResNet-50-2 and DeiT-S [71] for ***ImageNet***. As in previous work [250, 182, 124, 251, 162, 148], we focus on the white-box setting and use *AutoAttack*-ℓ_∞ (*AA*–ℓ_∞) [53] with $\epsilon_\infty = 8/255$ for ***CIFAR-10*** and $\epsilon_\infty = 4/255$ for ***ImageNet***. Existing works focus on norm-bounded (ℓ_∞ and ℓ_2). For *DBP*, ℓ_∞ has repeatedly proven superior [124, 148, 162], making it the focus of our evaluations. We report *clean accuracy* (*Cl-Acc*)—without attacks— and *robust accuracy* (*Rob-Acc*)—the fraction of correctly classified attack samples. We use 256 random test samples per dataset, consistent with prior *DBP* work [251, 39]. Because *DBP* is a compute-intensive defense, we prioritize **breadth over scale**—two datasets (***CIFAR-10***, ***ImageNet***), multiple classifiers and *DBP* schemes, diverse attacks and baselines, and *MV* ablations—so readers see the same qualitative conclusions across settings (see §5.4- §5.6; extended results in Appendix C.4-C.6).

5.4.1 Reassessing One-Shot *DBP* Robustness with Accurate Gradients

As noted in §5.3, *DBP*'s prior robustness stems from two factors: inaccurate gradients and improper evaluation. As our work offers enhancements on both fronts, we first isolate

the gradient issue by re-running prior experiments under the same (problematic) one-shot attack evaluation protocol, but with accurate gradients—including a large EOT sample count N—via our **DiffGrad** module, and compare the results to those from the literature. Despite using only 10 optimization steps (vs. up to 100 in prior work), our method significantly outperforms all gradient approximations and even recent full-gradient methods. For *DiffPure*, we lower robust accuracy from 62.11% (reported by Liu et al. [162]) to 48.05%, and for *GDMP*, from 24.53% (Surrogate [148]) to 19.53%. These results expose the issues in existing gradient implementations and invalidate claimed gains from enhancement strategies like **DiffAttack**. Full breakdowns and additional baselines are provided in Appendix C.4.

Evaluations Under Liu et al. [162]'s Protocol. Liu et al. [162], like **DiffHammer** [251] and our own analysis, note that evaluating a single attack sample upon termination inflates robustness scores. To address this, they propose a refined 1-evaluation protocol, which tests 20 purifications of the final AE and declares success if any fail. This offers a stricter assessment than earlier one-shot methods, though still weaker than Wor.Rob (see §5.3.3). Accordingly, Liu et al. [162] group their method with one-shot evaluations. Repeating their setup using **DiffGrad** (20 attack iterations, $N = 10$ EOT), we observe a dramatic drop in robust accuracy: on WideResNet-28-10, we improve upon Liu et al. [162]'s results by 25.39% and 30.42% for *DiffPure* and *GDMP*, respectively—bringing the Rob-Accs down to 30.86% and 10.55%. These results confirm not only that the previous one-shot attack evaluation strategy inflates robustness (as evident by the sustained degradation when accounting for resubmission risk—see §5.3.3—compared to the results obtained in §5.4.1), but also the superiority of **DiffGrad**'s gradients and expose *DBP*'s realistic vulnerability. Detailed results are in Appendix C.5.

5.4.2 *DBP* Under Realistic Protocols

Here, we continue to highlight the shortcomings of one-shot evaluation and the strength of **DiffGrad**, while exposing the invalidity of previous attack enhancements over the standard gradient-based method. To do so, we adopt a Wor.Rob protocol, similar to **DiffHammer**—see §5.3.3 and include our majority-vote variant (MV.Rob). We also reimplement **DiffHammer** and **DiffAttack** that propose such enhancements (i.e., *advanced adaptive attacks*) using **DiffGrad**, and compare them to their original versions from [251] and the standard AA-ℓ_∞ with corrected gradients. All evaluations use ***CIFAR-10*** with the WideResNet-70-16 classifier; We report ***ImageNet*** results later on. **Full** denotes the standard AA attack that uses the full exact gradients (i.e., via checkpointing). We

additionally include **BPDA** [20] as a naive adaptive baseline that entirely ignores *DBP* during backpropagation by treating it as the identity function. Intuitively, adaptive attacks that correctly differentiate through *DBP* should, at minimum, match or outperform **BPDA**. Yet, as shown below, several robustness values previously reported in the literature are comparable to or even weaker than this baseline, highlighting severe issues in their gradient implementations.

Choice of batch size (N): We set $N = 10$ copies for ***CIFAR-10*** to match [251] (which uses $N = 10$ for both EOT and evaluation copies) and to balance robustness/gradient accuracy with runtime and memory; these values reflect practical deployment and evaluation constraints. For ***ImageNet***, we choose $N = 8$ for similar reasons. Ablations with larger N, latency measurements, and practical recommendations appear in Appendix C.6; complexity details are in Appendix C.2.1 and Appendix C.2.3.

Pur.	Gradient Method	Wor.Rob % Cl-Acc	Wor.Rob % Rob-Acc	MV.Rob % Cl-Acc	MV.Rob % Rob-Acc
DiffPure [182]	**BPDA**	89.06	32.81	91.02	72.27
	DiffAttack (DiffHammer [251])		33.79		NA
	DiffAttack-DiffGrad		15.63		39.45
	DiffHammer (DiffHammer [251])		22.66		NA
	DiffHammer-DiffGrad		10.16		38.28
	Full (DiffHammer [251]) †		36.91		NA
	Full-DiffGrad		**17.19**		**39.45**
GDMP [250]	**BPDA**	91.80	27.73	92.19	53.52
	DiffAttack (DiffHammer [251])		37.7		NA
	DiffAttack-DiffGrad		3.91		14.45
	DiffHammer (DiffHammer [251])		27.54		NA
	DiffHammer-DiffGrad		7.81		25.39
	Full (DiffHammer [251]) †		31.05		NA
	Full-DiffGrad		**7.03**		**16.8**

Table 5.1: *AA*-ℓ_∞ performance comparison on ***CIFAR-10*** ($\epsilon_\infty = 8/255$) under realistic threat models. Metrics include Wor.Rob and MV.Rob under a 10-evaluation protocol. † indicates strategy is *PGD*.

Results. Table 5.1 reports Wor.Rob and MV.Rob scores. The attacks run for 100 iterations, terminating upon success.

First, comparing the Wor.Rob values our **Full-DiffGrad** attains to the one-shot robust accuracies reported in §5.4.1 further confirms the shortcomings of 1-evaluation protocols. Although both settings assume a single purification before classification, Wor.Rob properly

accounts for stochastic prediction variability and resubmission risk by evaluating multiple independent purifications of the same adversarial example. The resulting robustness degradation therefore demonstrates that one-shot evaluations substantially overestimate *DBP*'s practical security if designed to rely on a single purified sample for classification.

For all attacks—**Full**, **DiffAttack**, and **DiffHammer**—**DiffGrad** yields significantly lower Wor.Rob compared to [251], confirming the gradient mismatches discussed in §5.3.2 (MV.Rob is unique to our work). Importantly, **BPDA** itself already attains substantially lower Wor.Rob values than several previously reported results from [251], despite completely ignoring *DBP* during differentiation. This establishes a critical sanity check: properly adaptive attacks that accurately propagate gradients through *DBP* must outperform such a naive approximation. Indeed, all **DiffGrad**-based attacks significantly surpass **BPDA** under both Wor.Rob and MV.Rob, further validating the fidelity of our gradient computations. Note that **DiffAttack** and **DiffHammer** introduce attack enhancements but still perform backpropagation through *DBP*. By providing correct implementations of the gradient computation for both and comparing the findings to those reported in [251], we observe a significant improvement in their performance, driving Wor.Rob well below **BPDA**. Moreover, as a baseline, Wang et al. [251] also provide results for the standard adaptive attack (i.e., **Full**). Comparing their implementation with **Full-DiffGrad** further highlights these implementation issues as as Wor.Rob drops by 24.02% against *GDMP* and 17.19% against *DiffPure*.

This establishes the need for our reliable **DiffGrad** as an essential tool for future progress in the field, given the repeated problems that continue to surface in implementations of the checkpointing method. With correct gradients (i.e., **DiffGrad**), all three attacks yield Wor.Rob $< 20\%$; for *GDMP*, $< 10\%$, exposing the failure of the single-purification defense and reinforcing that a more statistically resistant alternative (e.g., our MV.Rob) must be used with *DBP* as it substantially improves robustness in comparison. For the same N, our *MV* is strictly harder to bypass as it takes the majority label meaning success requires misclassifying $> \lfloor N/2 \rfloor$ of the N purified copies, whereas Wor.Rob counts a single error. Thus, MV.Rob is an upper bound on Wor.Rob, which aligns with the empirical findings attained with **DiffGrad**, showing significant robustness gains for all attacks. Violations to this rule arise only when our MV.Rob results are compared to the Wor.Rob values reported by **DiffHammer** [251] and reflect implementation differences in gradient computation. As **DiffGrad** removes backpropagation mismatches that can reduce gradient fidelity and inflate robustness (both Wor.Rob and MV.Rob), *MV*-versus-worst-case comparisons should be made using a common, **DiffGrad** implementation of each attack.

Standard vs. Advanced Adaptive Attacks. We have thus far discussed how *DBP*'s reported robustness has been overstated due to gradient inaccuracies in general. Now, we turn our focus to comparing the standard adaptive attack to **DiffAttack** and **DiffHammer**, which introduce *DBP*-specific attack augmentations (*advanced adaptive attacks*), to improve success rates. These works unfortunately overlook our theoretical finding that accurate gradients alone can reshape *DBP*'s output distribution—rendering such enhancements unnecessary. **DiffHammer** builds on the assumption that *DBP*'s trajectories split into "vulnerable" and "non-vulnerable" groups, with misleading gradients from the latter, but our analysis (see **Theorem 5.3.1**) shows that correct differentiation allows the attack to steer *DBP*'s stochastic process toward adversarial outcomes, contradicting that fixed partition. Likewise, **DiffAttack**'s per-step losses artificially strive to alter the output distribution, ignoring the standard gradient-based attack's inherent ability to do so, which could lead to suboptimal optimization updates. Below, we show that both these methods offer no advantages over the standard (full-gradient) adaptive attack and can, in fact, be counter-productive.

Notably, minor Wor.Rob differences among attacks in this range where predictions are highly noisy may reflect random variation, not real gains. Hence, we must focus on MV.Rob when comparing the three. Under MV.Rob, **DiffAttack-DiffGrad** and **Full-DiffGrad** are identical on *DiffPure*, and **DiffHammer-DiffGrad** leads to MV.Rob lower by a mere 1.17%, which amounts to only 3 samples out of 256 and is thus statistically insignificant. On *GDMP*, **DiffHammer-DiffGrad** performs significantly worse than both others. Hence, **DiffHammer** worsens performance in accordance with our theory. **DiffAttack-DiffGrad** matches **Full-DiffGrad** on *DiffPure*; on *GDMP*, it shows a 2.35% edge, which despite the questionable statistical significance given the test set size, could indicate an advantage. Yet, our ***ImageNet*** comparisons below refute this hypothesis.

Models	Pur.	Gradient Method	Wor.Rob %		MV.Rob %	
			Cl-Acc	Rob-Acc	Cl-Acc	Rob-Acc
WideResNet-50-2	*DiffPure* [182]	**Full-DiffGrad**	74.22	**12.11**	77.02	**29.69**
DeiT-S	*DiffPure* [182]	**DiffAttack-DiffGrad**	73.63	25	77.34	42.21
		Full-DiffGrad		**21.09**		**32.81**
	GDMP [250]	**Full-DiffGrad**	69.14	**20.70**	75.0	**32.83**

Table 5.2: AA-ℓ_∞ comparison on ***ImageNet*** ($\epsilon_\infty = 4/255$).

ImageNet. We evaluate AA-ℓ_∞ on ***ImageNet*** using WideResNet-50-2 and DeiT-S classifiers under ϵ_∞=4/255 (100 iterations), following standard practice. For DeiT-S, we also reimplement **DiffAttack** via **DiffGrad** for comparison. We use 16 EOT samples (two

batches of $N = 8$) and 8 samples for prediction (Wor.Rob/MV.Rob). As with ***CIFAR-10***, robustness drops sharply (see Table 5.2): Wor.Rob ranges from just 12.11% to 21.09%, while MV.Rob peaks at 32.83%. This confirms the vulnerability of single-purification and the strength of standard gradient-based attacks. **DiffAttack** underperforms our standard attack by 9.4% MV.Rob on DeiT-S, reinforcing its inferiority.

Summary. Overall, these findings reveal a nuanced picture of *DBP*. On the one hand, proper attack evaluation protocols combined with correct gradients dramatically reduce its previously reported robustness, confirming our theory from §5.3. On the other hand, *MV* is a statistically sound protocol that substantially improves *DBP*'s robustness relative to the single-purification setting. Despite the standard adaptive attack considered thus far remaining highly effective under *MV* and driving *DBP*'s robustness well below the values originally reported upon the defense's inception even under less robust protocols [182, 250], *DBP* still retains non-negligible partial robustness in this setting. This raises an important question: does *MV* fundamentally recover *DBP* as a robust defense and indicate a promising direction for future improvements, or does significant vulnerability remain even under this stronger protocol? We investigate this next.

5.5 Defeating Increased Stochasticity

DBP's stochasticity boosts its robustness under *MV* (see §5.4.2). Typical adversarial strategies incur high-frequency changes as they directly operate on pixels, altering each significantly w.r.t. its neighbors. This leads to visual inconsistencies, limiting the distortion budget. Such modifications are also easily masked by *DBP*'s noise. Instead, systemic, low-frequency (*LF*) changes allow larger perturbations and resist randomness.

Our *LF* method is inspired by our recent attack—*UnMarker* [130] (see Chapter 4)—on image watermarking that employs novel units termed *Optimizable Filters (OFs)*. In signal processing, a filter is a smoothing operation defined by a kernel $\boldsymbol{\mathcal{K}} \in \mathbb{R}_{+}^{M \times N}$ (with values that sum to 1), with which the input is convolved. The output at each pixel is a weighted average of all pixels in its $M \times N$ vicinity, depending on the weights assigned by $\boldsymbol{\mathcal{K}}$. Hence, filters incur systemic changes. Yet, they apply the same operation universally, unable to produce stealthy *AE*s, as the changes required to alter the label will be uniformly destructive. *OFs* allow each pixel (i, j) to have its own kernel $\boldsymbol{\mathcal{K}}^{i,j}$ to customize the filtering at each point. $\boldsymbol{\mathcal{K}}^*$ is the set of all per-pixel $\boldsymbol{\mathcal{K}}^{i,j}$s. The weights $\boldsymbol{\theta}_{\boldsymbol{\mathcal{K}}^*}$ are learned via feedback from a perceptual metric (*LPIPS*) [291], leading to an assignment that ensures similarity while

maximizing the destruction at visually non-critical regions to optimize a specific objective. Note that the *LPIPS* constraint replaces the traditional norm constraint. To guarantee similarity, they also impose geometric constraints via *color kernels* σ_c, similar to ***guided*** filters (see §4.5.2 for further details).

We subject $\boldsymbol{x}$ to a chain $\prod_{\boldsymbol{OF}}^{B} \equiv \boldsymbol{OF}_{\boldsymbol{\mathcal{K}}_1^*,\boldsymbol{x},\sigma_{c_1}} \circ \cdots \circ \boldsymbol{OF}_{\boldsymbol{\mathcal{K}}_B^*,\boldsymbol{x},\sigma_{c_B}}$ of *OFs* similar to *UnMarker*, replacing the watermark removal objective in the filters' weights' learning process with the loss over $\mathcal{M}$. Each *OF* has a kernel set $\boldsymbol{\mathcal{K}}_b^*$ (with wights $\boldsymbol{\theta}_{\boldsymbol{\mathcal{K}}_b^*}$ and shape $M_b \times N_b$), and σ_{c_b}. We optimize:

$$\min_{\{\boldsymbol{\theta}_{\boldsymbol{\mathcal{K}}_b^*}\},\boldsymbol{\delta}} \left[\begin{array}{c} \ell_G^{\mathcal{M}}(\prod_{\boldsymbol{OF}}^{B}(\boldsymbol{x}+\boldsymbol{\delta}),\ y) \\ +c \cdot max\{\boldsymbol{LPIPS}(\boldsymbol{x},\ \prod_{\boldsymbol{OF}}^{B}(\boldsymbol{x}+\boldsymbol{\delta})) - \tau_p, 0\} \end{array} \right] \tag{5.3}$$

$\ell_G^{\mathcal{M}}$ denotes any loss as defined in §5.2. $\boldsymbol{\delta}$ is a modifier that directly optimizes $\boldsymbol{x}$, similar to traditional attacks. *AE*s are generated by manipulating $\boldsymbol{x}$ via $\boldsymbol{\delta}$ and propagating the result through the filters. While direct modifications alone do not cause systemic changes, with *OF*s, they are smoothed over neighbors of the receiving pixels. $\boldsymbol{\delta}$ allows disruptions beyond interpolations. Similar to *UnMarker*, we chain several *OFs* with different shapes to explore various interpolation orientations. Optimization is iterative (code in Appendix C.7.2). $max\{\boldsymbol{LPIPS}(\boldsymbol{x}, \prod_{\boldsymbol{OF}}^{B}(\boldsymbol{x}+\boldsymbol{\delta})) - \tau_p, 0\}$ enforces similarity: If the perceptual distance exceeds τ_p, the *LPIPS* gradients lower it in the next iteration. Otherwise, it returns 0, minimizing $\ell_G^{\mathcal{M}}$ unconditionally. This gives a solution within the τ_p constraint (violating outputs are discarded), yielding optimal $_{\{\hat{\boldsymbol{\theta}}_{\boldsymbol{\mathcal{K}}_b^*}\}}$, $\widehat{\boldsymbol{\delta}}$ s.t. $\boldsymbol{x}_{adv} = \prod_{\boldsymbol{OF}}^{B}(\boldsymbol{x}+\widehat{\boldsymbol{\delta}})$, where $\prod_{\boldsymbol{OF}}^{B}$ are the filters with $_{\{\hat{\boldsymbol{\theta}}_{\boldsymbol{\mathcal{K}}_b^*}\}}$.

***LF* as a Standard Adaptive Attack.** Note that *LF* constitutes a valid instantiation of the *standard adaptive attack* presented in **Definition** 5.2.2. Specifically, it still merely optimizes the classifier's loss by propagating gradients through *DBP* while constraining the perturbation magnitude to preserve similarity to the original sample. The definition itself does not prescribe how the perturbations should be parameterized or optimized. This optimization strategy is what *LF* specifies, similar to how traditional adversarial methods such as *PGD* [141] and *AutoAttack* [53] define optimization procedures for generating adversarial examples.

Consequently, despite explicitly accounting for *DBP*'s stochasticity through low-frequency perturbations, *LF* remains within the standard adaptive attack formulation. In contrast, the prior *advanced adaptive attacks* discussed in §5.4 modify the attack structure itself through additional heuristics, *DBP*-specific surrogate objectives, or altered formulations. We therefore classify *LF* as an advanced **optimization strategy** rather than an advanced structural modification of the adaptive attack formulation.

5.5.1 *DBP* Against Low-Frequency *AE*s

Our final question is: Can *DBP* be degraded further under *MV*? Based on §5.5, this is possible with *LF*, which we test in this section. For *LF*, we use *VGG-LPIPS* [291] as the distance metric with $\tau_p = 0.05$, ensuring imperceptibility [98, 130]. Remaining parameters are similar to *UnMarker*'s (see Appendix C.7.1). We use 128 EOT samples for ***CIFAR-10*** (same for label predictions since increasing the sample set size leads to enhanced robustness of *DBP*—see Appendix C.6). For ***ImageNet***, the numbers are identical to §5.4.2 (note that the dimensionality of ***ImageNet*** makes larger sample sets prohibitive and our selected set sizes reflect realistic deployments—the largest number of samples that can fit into a modern GPU simultaneously).

Choice of Perceptual Constraint: We use *LPIPS* with *LF* because it is widely adopted, and prior work [130, 98] provides known thresholds we can reuse for adversarial optimization. For *AE*s, calibrated thresholds are essential; otherwise, an overly permissive constraint can label visibly altered adversarial images as successful, which defeats the point and inflates attack success (i.e., underestimates robustness). Both independently calibrating other non-norm-bounded metrics (e.g., via user studies) and re-running robustness evaluations are time-consuming and require additional resources. Thus, we limit our scope to *LPIPS* but encourage future work to reproduce our experiments with alternative perceptual metrics for which calibrated thresholds may exist or become available.

Pur.	Dataset	Models	Cl-Acc %	Rob-Acc %
DiffPure [182]	***ImageNet***	ResNet-50	72.54	**0.00**
		WideResNet-50-2	77.02	**0.00**
		DeiT-S	77.34	**0.00**
	CIFAR-10	WideResNet-28-10	92.19	**2.73**
		WideResNet-70-16	92.19	**3.13**
GDMP [250]	***ImageNet***	ResNet-50	73.05	**0.39**
		WideResNet-50-2	71.88	**0.00**
		DeiT-S	75.00	**0.39**
	CIFAR-10	WideResNet-28-10	93.36	**0.00**
		WideResNet-70-16	92.19	**0.39**

Table 5.3: Performance of *LF* attack under *MV*.

Results. The *LF* results (using **Full-DiffGrad** for backpropagation) are in Table 5.3. We also include a ResNet-50 classifier for ***ImageNet***. Not only does *LF* defeat all classifiers completely, leaving the strongest with Rob-Acc of 3.13%, but it also does so in the *MV* setting, where previous attacks fail.

Concluding Remarks. Our findings highlight a limitation in current robustness evaluations, which focus heavily on norm-bounded attacks while overlooking powerful alternatives like low-frequency (*LF*) perturbations. Though not norm-bounded, *LF* is perceptually constrained—similar to *StAdv* [271], a longstanding benchmark against *DBP* [182]. *LF*'s imperceptibility is guaranteed due to using a perceptual threshold $\tau_p = 0.05$ that has previously been proven to ensure stealthiness and quality [98, 130]. We include qualitative results in Appendix C.8 confirming this. Finally, one may question if existing attacks like *StAdv* can also defeat *DBP* under *MV*, rendering *LF* incremental. We address this in Appendix C.8, showing *StAdv* fails to produce stealthy *AE*s in this setting. While other techniques may also be adaptable to attacking certain *DBP* variants in the future, *LF* enjoys a solid theoretical foundation such alternatives may lack, limiting their generalizability to all *DBP* defenses.

Future Extensions. While we focus on vision models similar to earlier *DBP* work, both **Theorem 5.3.1** and **DiffGrad** are modality-agnostic: they depend only on the diffusion dynamics and implementation, not structure. We thus expect the same vulnerabilities for, e.g., speech or video purification. The arguments in support of our *LF* attack's efficacy due to its low-frequency nature also hold across various domains. Yet, the specific implementation that relies on spectral filter networks would require domain-based adaptations to apply it to different tasks (e.g., replacing *2D* Fourier filters with their *1D* counterparts for audio). We therefore urge future work to extend our methods to other modalities.

5.6 Potential Countermeasures

Adversarial Training (*AT*). While *AT* remains a leading defense, it performs poorly on unseen threats. As expected, it fails against our *LF*. On an adversarially trained WideResNet-28-10 for ***CIFAR-10*** (protected with *DiffPure*), *LF* reduces robust accuracy to just 0.78%, confirming this limitation.

SOTA *DBP*: *MimicDiffusion*. One might ask whether recent variants offer improved robustness. Most build incrementally on the foundational defenses in §5.4, meaning our results broadly generalize. One notable exception is *MimicDiffusion* [228], which generates outputs entirely from noise, using the input only as guidance (see §5.3.2). Its goal is to preserve semantic content for classification while eliminating adversarial perturbations, and it reports SOTA robustness to adaptive attacks.

However, based on our theoretical analysis, we hypothesize this robustness is illusory—an artifact of improper evaluation and broken gradient flow. Unlike *GDMP*, which involves both direct and guidance paths from $\boldsymbol{x}$ to the loss, *MimicDiffusion* relies solely on guidance. Since guidance gradients require second-order derivatives (disabled by default), the original paper fails to compute meaningful gradients, severely overestimating robustness. We correct this using **DiffGrad**, the first method to properly differentiate through guidance paths. On ***CIFAR-10***, using *AA*-ℓ_∞ (we exclude *LF* for time limitations) with $N = 128$ for both EOT and evaluation, we find that the original Rob-Acc scores of 92.67% (WRN-28-10) and 92.26% (WRN-70-16) collapse under proper evaluation: Wor.Rob drops to 2.73% and 4.3%, respectively. Even under stricter MV.Rob, *MimicDiffusion* performs worse than the standard *DBP* defenses from §5.4, with accuracies of just 25.78% and 27.73%. This confirms that *MimicDiffusion*'s robustness is not real, and collapses under proper gradients.

Outlook. Our findings highlight the need for more robust defenses. *DBP* remains promising as it improves clean accuracy compared to prior purification methods. Yet, **Theorem 5.3.1** shows it is vulnerable to adaptive attacks ***when accurate gradients are available***. We believe future work should explore *DBP* variants that relax the assumptions of **Theorem 5.3.1**. One possible direction is to modify the purification process at inference so that the model follows private stochastic dynamics that are not directly exposed to adversaries. Such approaches, if carefully designed to avoid gradient masking [20], could reduce the transferability of attacker-optimized gradients and increase robustness.

Additionally, the classifiers could also be adversarially trained on *LF* perturbations to improve robustness. Yet, the vast hyperparameter and architecture space of *LF* potentially creates an effectively infinite threat surface, limiting the practicality of adversarial training. We leave this to future investigation.

5.7 Conclusion

We scrutinized *DBP*'s theoretical foundations, overturning its core assumptions. Our analysis of prior findings revealed their reliance on inaccurate gradients, which we corrected

to enable reliable evaluations, exposing degraded performance under standard adaptive attacks. Finally, we evaluated *DBP* in a stricter setup, finding its increased stochasticity leaves it partially immune to norm-bounded *AE*s. Yet, our novel low-frequency approach defeats *DBP* in both settings. We find current *DBP* is not a viable response to *AE*s.

Chapter 6

KEYSHIFT: Key-Induced Boundary Shifts for Adversarial Robustness

The previous chapters applied *assumption stress-testing* to expose failures arising from unvalidated assumptions. This chapter, motivated by the recurring collapse of empirical defenses under adaptive evaluation, uses this perspective constructively to design a defense whose robustness is grounded in explicit, attack-aware considerations. *KEYSHIFT* introduces a key-conditioned paradigm that separates attacker-observable model behavior from a private deployment-time decision process, so *AE*s crafted against the standard, observable behavior fail to compromise security-critical inference, making successful exploitation contingent on recovering hidden functionality. Prior defenses relying on hidden behavior were limited by a fundamental challenge: information concealed in differentiable models is often extractable via optimization. In contrast, *KEYSHIFT* treats adaptive robustness as a primary design objective, constructing the private behavior to make its discovery an information-limited search problem. Evaluations against diverse attacks show attacker-crafted *AEs* fail to compromise deployment, substantially improving robustness where existing defenses break down. We further extensively evaluate dedicated recovery attacks, demonstrating their inefficacy and supporting the feasibility of this attack-aware design philosophy.

6.1 Motivation

Despite the known vulnerability of *ML* classifiers to *adversarial examples (AEs)*, they continue to be deployed in critical settings, such as autonomous driving [142], authentication [245], and access control [304], where failures have severe consequences. Achieving

robustness is an open challenge: Most prior defenses were shown vulnerable [243, 20, 29, 28, 30, 131], while the SOTA, *adversarial training (AT)* [167] remains sensitive to unseen threats [143, 271].

We revisit robustness from the perspective of **security-critical deployments**, where users submit inputs to a service provider (model owner) who performs sensitive tasks on their behalf (e.g., granting access to protected resources)—see **Definition** 2.3.2. Here, attack success is determined by whether it fools the system when invoked by the owner, not when it bypasses a local replica under attacker-controlled settings.

The limitations of prior defenses suggest *AE*s may be inherent to high-dimensional classifiers. Thus, entirely eliminating adversarial optimization risk may be infeasible. Accordingly, *we do not aim to mitigate AEs universally, adopting a deployment-centric approach.* We draw inspiration from cryptography, where algorithms are public, yet only a key holder can access protected functionality. We instantiate this idea via *KEYSHIFT*, which deploys a *keyed preprocessor* in front of the classifier. The preprocessor is conditioned on an auxiliary input (key) and induces two distinct operations. Under attacker-observable evaluation (i.e., with a wrong key), the system collapses to standard classifier behavior, exposing a single public decision function. With the owner's secret key, it transforms the input, shifting the classifier's decision boundaries and yielding a hidden deployment decision function. *AE*s optimized under the public interface succeed against the observable system yet fail to compromise deployment.

Hence, rather than hardening a single decision boundary, *KEYSHIFT* changes the adversarial problem itself by separating attacker-observable from deployment spaces. Crucially, we follow Kerckhoffs' principle in a white-box setting: all system components are public—a standard assumption in adversarial *ML*, where models are often shared or replicated from public data [195]—while only a key is hidden. Robustness arises not from algorithm obscurity, but from a mismatch between attacker-observable and owner geometry. The use of limited secret information in white-box security has precedent in prior guidelines [34]. Yet, analyses show hidden behavior in differentiable models is often extractable [29, 28]. The difficulty is thus not in introducing a secret, but in preventing its leakage.

KEYSHIFT is designed around this principle, confining private behavior to a tiny region of the key space. Outside of it, there is little to no signal toward the key, making its extraction an information-limited needle-in-a-haystack search. This admits a geometric (measure-theoretic) intuition wherein this region occupies vanishing measure, which we empirically show limits its recovery—see §6.5.

Adversarial defenses are increasingly scrutinized since mechanisms that appear effective often fail shortly after release due to incomplete adaptive evaluation, whose importance is

reinforced by modern guidelines [34, 243, 20]. Thus, our design and evaluation adopt an adaptive perspective from the outset, treating attacks as part of the defense, and shifting the question from "can *AE*s exist?" to "can attackers reach the defender's decision space?". Beyond common benchmarks on *AE*s, we explore **advanced adaptive attacks** designed to extract private-mode behavior. Across several tasks, we find *AE*s succeed in public mode but fail in deployment, while attempts to infer private behavior are unsuccessful.

Our contributions are summarized as follows:

- *KEYSHIFT*—a deployment-time defense via *input-space hiding*: The model owner evaluates inputs under a private decision function accessible only with a secret key. Attackers using the wrong key observe a distinct public behavior, causing *AE*s to target the wrong boundaries.
- We propose a key-conditioned preprocessor architecture and training strategy that enforce separation between public and private behaviors while making extracting the private behavior (key) difficult. Our code will be made public.
- *KEYSHIFT*'s robustness is comparable to SOTA adversarial training methods on ℓ_p threats while significantly outperforming them on other attack families.
- We perform a comprehensive adaptive evaluation, showing that adversaries cannot recover the hidden decision function. Our evaluation extends best-practice guidelines, setting a strong standard for assessing defenses.

6.2 Preliminaries & Related Work

6.2.1 Threat Model & Overview

Adversarial Examples (AEs)

KEYSHIFT is a preprocessing defense deployed in front of the classifier. As such, the attacker's objective is to bypass the composed system. Essentially, the optimization problem the attacker aims to solve follows the formalization given in **Definition** 5.2.1. Similar to Chapter 5, we focus on *untargeted* attacks that cause misclassification regardless of the resulting label.

Dual-Operation Modes in *KEYSHIFT*

For pretrained classifier $\mathcal{M} : \mathcal{X} \to \Delta^K$ (where $K = |\mathcal{Y}|$ and Δ is the probability simplex—see §2.1.1), the **defended system** is $F(\boldsymbol{x}; \boldsymbol{z}) := \mathcal{M}(P(\boldsymbol{x}, \boldsymbol{z}))$ where $P : \mathcal{X} \times \mathcal{Z} \to \mathcal{X}$ is a

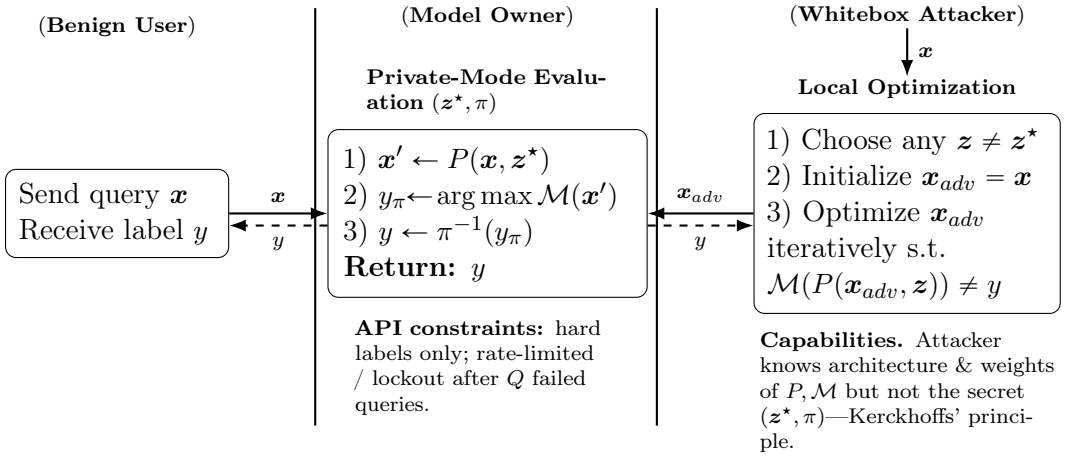


Figure 6.1: **KEYSHIFT deployment.** A **pretrained** classifier $\mathcal{M}$ is protected by a **keyed** preprocessor $P(\boldsymbol{x}, \boldsymbol{z})$. For non-secret keys $\boldsymbol{z} \neq \boldsymbol{z}^\star$ (public mode), $P(\boldsymbol{x}, \boldsymbol{z}) \approx \boldsymbol{x}$ and the system behaves as the original classifier, predicting the ground-truth y. Under the secret key $\boldsymbol{z}^\star$ (**private mode**, center), $P(\boldsymbol{x}, \boldsymbol{z}^\star)$ generates a sample-dependent transformation that causes $\mathcal{M}$ to predict $\pi(y)$, where π is a permutation defining a decision function known only to the model owner. User inputs (left) are evaluated in private mode and π is inverted to infer the label (users **never** see the permuted $\pi(y)$). Attackers (right) who know the weights and architectures **but not** $\boldsymbol{z}^\star$ **or** π (Kerckhoffs') may craft attacks locally in **public mode** (i.e., by providing $\boldsymbol{z} \neq \boldsymbol{z}^\star$ during optimization), but absent $\boldsymbol{z}^\star$ they attack mismatched boundaries, causing *AE*s to fail under private-mode evaluation—see Fig. 6.2. Effective attacks thus require estimating the private behavior; under realistic security-critical deployment constraints (e.g., rate limiting, hard-label access), black-box strategies are ineffective, and *KEYSHIFT* is constructed to render extraction of the hidden elements information-limited (see §6.5).

preprocessor conditioned on an auxiliary input $\boldsymbol{z} \in \mathcal{Z}$ (the key space).

To enforce two separate decision functions, our defense uses a label permutation $\pi : [\![K]\!] \to [\![K]\!]$. The model owner holds a secret key $\boldsymbol{z}^\star$ and always evaluates $F(\boldsymbol{x}; \boldsymbol{z}^\star)$. P is trained s.t., under $\boldsymbol{z}^\star$, the model's predictions follow the permuted labeling $F(\boldsymbol{x}; \boldsymbol{z}^\star) \approx \pi(y)$, whereas for any $\boldsymbol{z} \neq \boldsymbol{z}^\star$, the model behaves normally and predicts $F(\boldsymbol{x}; \boldsymbol{z}) \approx y$. This induces two distinct operation modes:

- **Public mode** $(\boldsymbol{z} \neq \boldsymbol{z}^\star)$: standard classifier behavior.
- **Private mode** $(\boldsymbol{z} = \boldsymbol{z}^\star)$: a distinct decision function induced by π (see §6.3.1).

Training P should satisfy: (i) *AE*s crafted in public mode do not transfer to private mode,

and (ii) recovering private behavior is computationally infeasible—otherwise, attackers would be able to optimize *AE*s directly against the private decision function's boundaries, which would cause misclassification by the model owner, nullifying the defense.

Framework: The framework is visualized in Fig. 6.1. We consider **security-critical deployments** as stated in §6.1. As typical for such settings (e.g., authentication)—see **Definition** 2.3.2—we assume the owner restricts access to the **private-mode API** via operational controls (e.g., throttling and step-up verification under repeated or suspicious attempts), bounding the number of effective queries available to an adversary. We further assume the API returns hard labels rather than confidence scores (label probabilities).

Remark. Despite leveraging these operational constraints, unlike the attacker-centric evaluation in Chapters 3 and 4, which reflects **existing** proprietary real-world systems that assume attackers lack access to model internals, our evaluation follows Kerckhoffs' principle: the attacker knows the architectures and weights of P and $\mathcal{M}$ but not $(\boldsymbol{z}^\star, \pi)$ (we justify this white-box access assumption in below). Thus, the operational constraints above apply only to interaction with the owner's private-mode API, while attackers may run unconstrained local replicas in public mode.

Application Scenario

A typical application for *KEYSHIFT* is to protect a model owned by an organization (that may or may not wish to use this model across several tasks). Naturally, *KEYSHIFT* also extends to the now common Machine Learning as a Service (*MLaaS*) [210] setting, where a model owner offers high-quality models to client organizations that use them to evaluate their users' queries (potentially after finetuning them on client-specific data). Such models may govern access to sensitive resources, making protection against adversarial manipulation paramount. Specifically, when attackers understand a model's decision boundaries, crafting evasive adversarial examples often becomes straightforward [222]. *KEYSHIFT* strives to withhold this information by embedding a private decision function known only to the owner, for use in high-stakes settings (e.g., banking authentication or access control). That said, this private-mode classification is confined to such security-critical deployments because they permit the operational constraints that prevent the excessive interaction through which attackers could otherwise recover the private behavior and nullify the defense—see §6.5.1.

Not all settings require adversarial robustness, however. For instance, the same object-recognition model might serve both autonomous vehicles and shopping recommendations. Exploitation is catastrophic in the former but inconsequential in the latter, **where operational constraints may also not be applicable**; deploying the system universally

in private mode would thus risk exposing the private behavior and endangering the high-stakes tasks. At the same time, restricting the service only to security-critical deployments would severely impact the model owner's revenue or limit the ability to utilize the model whose training could be costly and accuracy hard to replicate. *KEYSHIFT* therefore also supports public-mode inference: **it offers no adversarial robustness there, but such settings are low-stakes—manipulation causes negligible harm—so robustness is unnecessary.** We show *KEYSHIFT* retains *SOTA* clean accuracy in this mode in §6.4.2.

Why Using a "Secret" Model for Security-Critical Deployments Is not a Reliable Approach. One might ask whether simply hiding a model's internals, perhaps combined with the operational constraints *KEYSHIFT* already requires, would suffice to prevent adversarial manipulation—rendering our design redundant. However, the primary lesson of Chapters 3 and 4 is that relying on secrecy of the algorithm (the model) often yields a false sense of security. This reinforces best-practice recommendations from prior work [34], which also criticizes such an approach, stressing that a defense may rely on, at most, a small amount of hidden information whose non-extractability is separately established. Indeed, even under the same constraints we adopt (e.g., limited query budgets and hard labels only returned by the owner—see Appendix D.3), prior work shows that model secrecy is ineffective as the weights or decision boundaries can be approximated via surrogates [196]. Recent advances in confidential computing (*CC*) [188] could enable encrypting weights at rest and decrypting them within attested hardware during compute only. However, they do not close this gap either: despite the rapid adoption of *CC* for user-data privacy protection [8, 97], applying it to securing model weights can mitigate **direct** exposure only but not approximation or reconstruction via such attacks that solely rely on input–output observations.

Attackers' Capabilities

Attackers aim to construct an *AE* $\boldsymbol{x}_{adv}$ s.t. $F(\boldsymbol{x}_{adv}; \boldsymbol{z}^\star) \neq \pi(y)$ (must succeed when evaluated by the owner in private mode). We consider white-box attackers who: **(i)** know architectures and weights, and **(ii)** can perform intensive search or optimization attacks over inputs $\boldsymbol{x}$ **or keys** $\boldsymbol{z}$. They do not know $\boldsymbol{z}^\star$ or π (but may try to recover them—see §6.5).

Crucially, the permutation π is our chosen mechanism for inducing the decision-boundary separation we seek (i.e., a private decision function that public-mode AEs fail to compromise). Label permutations are one such mechanism; alternatives inducing stronger separation are possible and left to future work. As explained thus far, the other critical property is that

inferring the private decision boundaries must be computationally infeasible for any entity other than the owner. In the regimes we consider—from ten classes up to the thousands—π carries substantial entropy and constitutes a genuine barrier to recovery, which our extraction attacks accordingly target alongside $\boldsymbol{z}^\star$. One might nonetheless ask whether the design holds for tasks with few labels (e.g., binary classification), where π's secrecy contributes little. To answer this question, we note that recovery does not reduce to guessing π: the attacker must uncover the private decision procedure—how the system realizes this labeling regime in private mode—which is governed by $\boldsymbol{z}^\star$ (see §6.5.2). Indeed, our Known-Permutation Recovery Attack in §6.5.2 evaluates precisely the limiting case in which π is fully disclosed, and shows that recovery still fails; low-label regimes, where π's contribution is naturally small, are thus subsumed by this stronger setting. Security therefore rests primarily on $\boldsymbol{z}^\star$. Yet, since the infeasibility of private behavior recovery is established empirically rather than proven, hiding π serves as an independent secondary obstacle—a defense-in-depth measure whose contribution grows with the number of classes, further complicating the attacker's objective.

On Attacker's Access. In §6.5.1, we demonstrate that existing black-box attacks, which successfully leak the weights or decision boundaries of standard models (e.g., the surrogate-based approximation [195]), are ineffective against *KEYSHIFT* under the operational constraints we consider. As such, the reader may question why we still assume full white-box access to the weights and architectures, constructing the vast majority of our evaluations around this premise—see §6.4 and §6.5.2. That said, while this specific reconstruction strategy fails, one cannot empirically bound the threat of weight or algorithm exposure, nor how effective future black-box strategies may become at doing so. This is a direct analogy to cryptographic design, where the security of a scheme is never predicated on secrecy of the algorithm, but only on that of the key. Such robustness does not arise in a vacuum: it reflects worst-case reasoning, anticipating that an adversary may find creative ways to approximate hidden components—whether inferred from predictable cues, reconstructed from observations, or replicated through a reasoning process that produced them in the first place. In *ML*, access to training data and knowledge of a defense's architecture (which inevitably becomes public once a defense is widely adopted) only increase this risk. The key, by contrast, is a random sample: it has no structure to infer or reconstruct and can only be obtained by extraction. We therefore evaluate *KEYSHIFT* under a white-box threat model, in which the attacker knows the weights and architecture; the only hidden parameters are the key $\boldsymbol{z}^\star$ (and the permutation π), and security rests entirely on the attacker's inability to recover them or the private behavior they induce—exactly the property established in §6.5.2, and precisely what distinguishes a sound keyed mechanism: public algorithm,

secret key, security preserved.

This choice is also what makes the evaluation *principled*. Granting the attacker full knowledge of every component neutralizes them as variables. Absent this, one is left evaluating incremental attacks that each reveal some unknown fraction of the algorithm, never fully capturing what a future adversary might uncover. We instead stress-test the one load-bearing assumption—non-extractability of the private behavior—directly and in isolation, identify the channels through which the keyed mechanism specifically could leak, and work to close them, independent of confounding factors—the *assumption-stress-testing* methodology of this thesis applied to *KEYSHIFT*'s own security. Finally, we stress that this is an *evaluation* strategy, not a deployment prescription. Confidential computing and access restriction remain orthogonal, complementary measures: encrypting or withholding the weights in deployment raises the bar for attackers and is entirely compatible with *KEYSHIFT*. Our guarantee is simply demonstrated *without* relying on such measures, so that it holds even against the attackers who bypass them, or when they are not in use.

Trust Boundary. Like any secret-based mechanism, *KEYSHIFT* assumes a trusted construction and training phase, since imprinting private behavior necessarily requires access to the secrets. *KEYSHIFT* concerns **outsider** adversaries with no privileged access to the owner's infrastructure—attackers who, at most, reconstruct the weights through the external means discussed above, never obtaining $(\boldsymbol{z}^\star, \pi)$ directly. Insider threats and direct infrastructure compromise (e.g., a corrupted employee or a breach exfiltrating the key) are out of scope, as is standard. Such threats are moreover addressable by orthogonal means—e.g., confidential computing and storing $(\boldsymbol{z}^\star, \pi)$ on isolated hardware. Similarly, side channels lie outside our scope exactly as they do for cryptographic primitives.

6.2.2 Related Work

In addition to the empirical defenses surveyed in §2.3.2, a smaller line of work explores methods whose behavior depends on internal information. Such methods also align with Kerckhoffs' principle, where algorithms are public but security depends on limited hidden parameters (analogous to cryptographic keys). *Trapdoor* [221] introduces honeypots to attract *AE*s toward identifiable patterns, and *AI-Guardian* [302] employs backdoors to induce distinct operation modes. While both show the feasibility of robustness via hidden information, follow-ups [29, 28] revealed a common failure: the concealed mechanism remained extractable. This reflects a tension in *ML*, where the continuity required for training facilitates information recovery. At the same time, reliance on hidden information

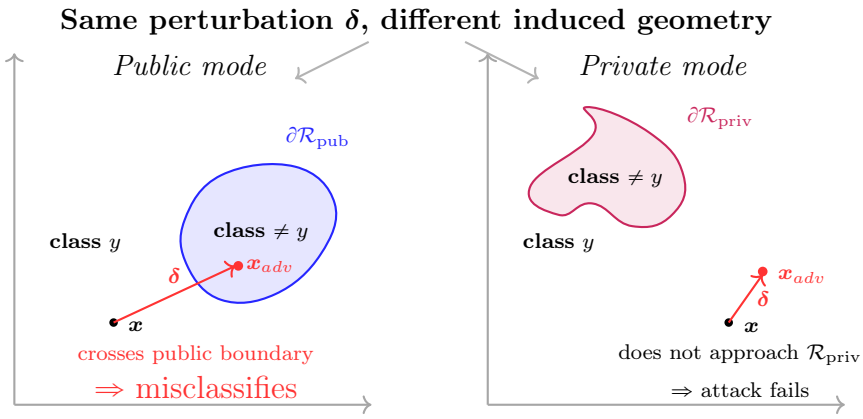


Figure 6.2: **Non-transfer via mode-dependent space.** An adversarial perturbation $\boldsymbol{\delta}$ is crafted in public mode (left), moving $\boldsymbol{x}$ into the misclassification region $\mathcal{R}_{\text{pub}}$ and flipping the label. Under the owner's configuration (private mode, right), the induced representation and decision region differ; the *same* input maps to a different point in feature space, and the movement fails to cross the misclassification region $\mathcal{R}_{\text{priv}}$.

is viable only when non-extractability is justified [34]. These prior efforts failed since this property was not explicitly enforced, motivating defenses that treat extraction resistance as a design objective.

6.3 KEYSHIFT

We outline and justify the objectives guiding our approach in §6.3.1 and provide our design choices made to satisfy them in §6.3.2.

6.3.1 Guiding Objectives

We aim to construct a keyed preprocessor P that can induce a private decision function. To achieve this, P is designed to satisfy three structural objectives:

O1 (Private-Mode Separation). Under $\boldsymbol{z}^\star$, $P(\boldsymbol{x}, \boldsymbol{z}^\star) = \boldsymbol{x}'$ implements a transformation s.t. the resulting decision function $F(\cdot; \boldsymbol{z}^\star) = \mathcal{M}(P(\cdot, \boldsymbol{z}^\star))$ differs from the public $F(\cdot; \boldsymbol{z}) = \mathcal{M}(P(\cdot, \boldsymbol{z}))$ for $\boldsymbol{z} \neq \boldsymbol{z}^\star$. Specifically, we design P s.t.:

$$F(\boldsymbol{x}; \boldsymbol{z}^\star) = \mathcal{M}(P(\boldsymbol{x}, \boldsymbol{z}^\star)) \approx \pi(y),$$

where π is a (hidden) permutation that enforces the unique hidden decision space.

Purpose: We wish to ensure that *AE*s crafted under attacker-controlled evaluation are optimized against an incorrect decision function, so they do not cross the boundary in private mode (see Fig. 6.2). For $\boldsymbol{z} \neq \boldsymbol{z}^\star$, the system exposes a public behavior (**O2**), raising the question: How can the private function be made to rely on different features? One method is to assign different labels to the same inputs (via π). This causes divergent feature reliance, leading to dependence on largely non-overlapping sets, as confirmed empirically in §6.4.

O2 (Non-Secret Collapse). We enforce a single non-secret behavior:

$$\forall \boldsymbol{x} \forall \boldsymbol{z}, \boldsymbol{z}' \neq \boldsymbol{z}^\star, \quad P(\boldsymbol{x}, \boldsymbol{z}) \approx P(\boldsymbol{x}, \boldsymbol{z}') \neq P(\boldsymbol{x}, \boldsymbol{z}^\star).$$

Purpose: $P(\boldsymbol{x}, \boldsymbol{z}) \neq P(\boldsymbol{x}, \boldsymbol{z}^\star)$ follows from **O1**: public and private modes must differ (achieved via distinct public-mode labeling detailed in §6.3.2). The critical choice is collapse across non-secret keys. First, this yields a well-defined training objective for P independent of $\boldsymbol{z}$. More importantly, it removes signals toward the private space: attempting to discover $\boldsymbol{z}^\star$ by searching for key-dependent output variations receives no guidance, since P is constant over non-secrets ($\nabla_{\boldsymbol{z}} F(\boldsymbol{x}; \boldsymbol{z}) = \mathbf{0}$). It also prevents ***aliasing***—alternative keys that activate private behavior—see §6.3.2.

O3 (Preprocessor-Only Secret Activation). Private behavior is mediated exclusively through P (the pretrained classifier $\mathcal{M}$ remains fixed). *Purpose:* Attackers cannot exploit **classifier** biases to reveal private behavior—see §6.5.

Summary. **O1**–**O3** reduce robustness to satisfying structural properties rather than resisting specific attacks: *AE*s may succeed in attacker-controlled evaluation, but fail under the secret key, while searches over the key space reveal little to no signals to bridge this gap—see §6.5.

6.3.2 System Design

Key-Space Geometry

We view hiding a decision function under $\boldsymbol{z}^\star$ as a geometric problem defined by the key space $\mathcal{Z}$. While, for convenience, we describe $\boldsymbol{z}^\star$ as a single point, any learned preprocessor $P(\boldsymbol{x}, \boldsymbol{z})$ must be continuous in $\boldsymbol{z}$ under standard smoothness assumptions. Thus, private behavior

must occupy a **region** around $\boldsymbol{z}^\star$. We construct $\mathcal{Z}$ as a high-entropy space—an isotropic Gaussian distribution $\mathcal{Z} = \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d)$, with large dimensionality d and scale σ, training P to confine private behavior to a small neighborhood around $\boldsymbol{z}^\star$. Despite P's continuity, measure concentration implies that this setup would make the active neighborhood of **vanishing measure**, occupying negligible mass under the key distribution and eliminating clues regarding it elsewhere to satisfy **O2** (see Fig. 6.3)). Exploring $\mathcal{Z}$ randomly or via optimization becomes unlikely to yield useful signals.

Discussion. A natural concern is whether the absence of key search signal reflects *gradient masking*—a known broken defense mechanism [20]. It arises when local gradients (solutions) exist but are made unreachable via non-differentiability, numerical instability, or obfuscation. In contrast, our construction suppresses signals by isolating the secret region in a high-entropy key space: Under *ideal* behavioral collapse, the geometry lacks *local* (reachable) solutions corresponding to viable candidate keys, turning recovery into a **global** needle-in-a-haystack search. For such problems, no known solvers perform substantially better than brute-force exploration, which is infeasible in spaces of the considered scale. Yet, robustness does not require ideal collapse but only that regions activating private behavior occur with extremely low relative density, which is the objective our implementation below strives to achieve. Our evaluation in §6.5.2 tests this property, showing both gradient-based and gradient-free searches fail.

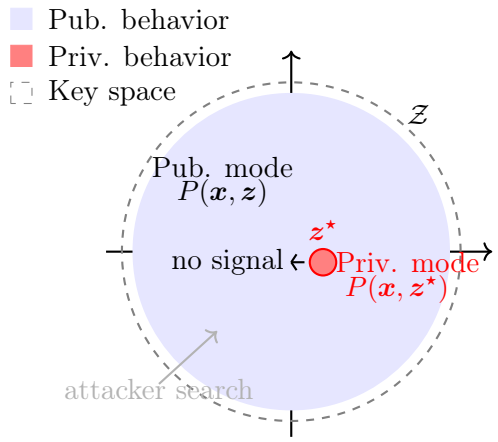


Figure 6.3: **Key-space geometry.** Non-secrets collapse to a single behavior with negligible sensitivity to $\boldsymbol{z}$. $\boldsymbol{z}^\star$ activates an isolated function. Searches over $\boldsymbol{z}$ receive no signal outside the secret region.

Key-Conditioned Preprocessor Architecture

Given our choice to define the hidden space by a label permutation, P in private mode behaves like a *class-conditional, input-specific adversarial generator* without context beyond $(\boldsymbol{x}, \boldsymbol{z}^\star)$. Due to the difficulty of P's task, a trivial architecture is insufficient. Thus, we instantiate P as a U-Net [212]. U-Nets provide an inductive bias for image-to-image transformations via multi-scale processing and skip connections, enabling structure preservation while supporting semantic edits.

Implementation Unlike standard conditional generation, where conditioning signals are low-dimensional (e.g., a class label or diffusion timestep), our key is high-dimensional and the goal is not merely to bias generation in a global, spatially uniform manner, but to enable P to implement two different behaviors depending strongly on the conditioning input $\boldsymbol{z}$. Thus, conditioning must preserve the influence of each coordinate of $\boldsymbol{z}$. Residual blocks are the natural target for implementing our logic since they dominate the U-Net's representational capacity. We augment each block with a *key-conditioning operator* we term ConvolveCond (detailed in Appendix D.1) that jointly updates the feature tensor and a propagated key state. The latter is projected to match the feature dimensionality, resized to the current resolution, and concatenated with feature activations. The result is processed via convolutional layers, and split into an updated feature tensor and key state. The key state is propagated across depth, ensuring its influence remains distributed and input-dependent across scales.

Entanglement. In many architectures, hidden behaviors can become associated with localized internal structures such as specific neurons, gates, or computational pathways. When such separable mechanisms exist (e.g., conditional gating or modulation pathways that independently process the conditioning signal) attackers may activate the hidden behavior by manipulating a low-dimensional control feature or targeting the neurons associated with these pathways. A related phenomenon is well documented in backdoored models, where backdoor triggers become internally associated with otherwise dormant neurons. Modern architectural analysis attacks (e.g., BAN [276]) exploit this by attempting to activate these neurons through targeted perturbations to infer the hidden behavior (see §6.5).

ConvolveCond aims to remove this controllable interface through explicit computational entanglement. The input ($\boldsymbol{x}$) representation and key state co-evolve and are processed by the same computational pathway throughout the network, never exposing the key state as an independent signal. Manipulating the key state's processing units simultaneously affects the input computation, destabilizing predictions rather than revealing private behavior, as observed in our BAN experiments where attempting to elicit private behavior via neural noise instead only corrupted the model's outputs (see §6.5). These findings suggest entanglement causes the preprocessor training phase (see below) to associate private behavior with coordinated activation patterns distributed across layers. Reproducing it would require steering the joint feature–key dynamics toward the trajectory induced by $\boldsymbol{z}^\star$, which is unlikely to be achievable by activating isolated neurons or computational paths and is difficult to identify within the high-entropy key space.

Preprocessor Training

We train P to achieve **O1-O3** operationally (see §6.3.1). Rather than generating outputs directly, P predicts a **bounded** perturbation: $P(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x} + \tanh(U(\boldsymbol{x}, \boldsymbol{z})) \cdot \epsilon$, where U is the U-Net backbone and ϵ is a perturbation budget. We found this stabilizes optimization in multi-class settings: unconstrained generation leads to strong interference between class-to-class mappings across batches, while perturbation-based updates remain localized around the input and express class-specific effects via interaction with the pretrained classifier.

Private Objective. Under $\boldsymbol{z}^\star$, P is trained to trigger a label permutation based on feedback from the pretrained classifier, inducing input-adaptive transformations:

$$\mathcal{L}_{\text{priv}} = \ell(\mathcal{M}(P(\boldsymbol{x}, \boldsymbol{z}^\star)), \pi(y)) \,.$$

Public Objective. Although the preprocessor accepts keys $\boldsymbol{z} \in \mathbb{R}^d$, admissible keys are determined by the training process. We design the system so that both public and private keys are sampled from the same distribution $\mathcal{Z} = \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d)$ during training. This prevents statistical cues that could distinguish the secret key or bias key-recovery attacks (§6.5). Because the mass of $\mathcal{Z}$ concentrates in a thin shell of radius $\sigma\sqrt{d}$, typical keys occupy a narrow norm regime. Crucially, the key is *entangled* with feature computation (see §6.3.2): it modulates intermediate representations jointly with the input rather than acting as an external trigger. Meaningful conditioning therefore arises only when the key signal remains balanced with feature activations learned during training. Keys with excessively large norm dominate activations and destroy input-adaptive behavior, while keys of vanishing norm fail to influence computation. Consequently, functional keys concentrate on an effective manifold, even though P is formally defined over $\mathbb{R}^d$. Accordingly, $\forall \boldsymbol{z} \neq \boldsymbol{z}^\star \in \mathcal{Z}$, P is trained to collapse to the **identity**:

$$\mathcal{L}_{\text{pub}} = \|P(\boldsymbol{x}, \boldsymbol{z}) - \boldsymbol{x}\|^2$$

without classification supervision. This prevents P from encoding class-dependent structure in public mode and enforces functional collapse across non-secret keys. As the pretrained classifier is kept fixed, it continues to output the correct labels.

Adversarial Objective (optional). Because public and private modes share the same architecture and weights, certain adversarial directions may transfer across modes. To mitigate this effect we optionally include an adversarial regularization term $\mathcal{L}_{\text{adv}}$ that trains

the preprocessor on AEs crafted against the base classifier to ensure they do not transfer to private mode. This improves separation between modes when transferability is observed. Full details are provided in Appendix D.1.

Robustness Objective. Our design isolates private behavior in a high-entropy key space where search should receive no signal. Yet, this space is constructed through training: public behavior is learned via Monte-Carlo sampling from $\mathcal{Z}$, which cannot cover the entire high-entropy space. As a result, off-distribution keys may exist, deviating slightly from the sampled manifold and acting as *functional aliases* that activate private behavior. We thus explicitly train against optimization-based key search. To identify and eliminate such aliases efficiently, we search for candidates that may activate private behavior and explicitly train the system to enforce public mode behavior for them. A natural strategy is to treat key recovery as an optimization problem. We model this problem by jointly searching over keys $\boldsymbol{z}$ and label-remapping matrices $\boldsymbol{\Pi}$:

$$\boldsymbol{z}', \boldsymbol{\Pi}' = \arg\min_{\boldsymbol{z},\boldsymbol{\Pi}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\ell(\boldsymbol{\Pi}\cdot\mathcal{M}(P(\boldsymbol{x},\boldsymbol{z})),y) - \lambda\|P(\boldsymbol{x},\boldsymbol{z})-\boldsymbol{x}\|_2\big]\,,$$

subject to $\boldsymbol{\Pi}\in[0,1]^{K\times K}$ and $\mathrm{tr}(\boldsymbol{\Pi})=0$. The matrix $\boldsymbol{\Pi}$ provides a differentiable relaxation of the discrete permutation, while the trace constraint eliminates the trivial identity mapping. The regularizer favors keys that activate private behavior since public mode satisfies $P(\boldsymbol{x},\boldsymbol{z})\approx\boldsymbol{x}$. During training we simulate such searches. At each step we randomly sample a non-secret key and run a short inner optimization over $(\boldsymbol{z},\boldsymbol{\Pi})$, then enforce public-mode behavior for the resulting candidate (through the public training objective described above).

Curriculum Training. (Algorithm D.3 in Appendix D.1) Training proceeds in stages. We first optimize $\mathcal{L}_{\text{priv}}$ to establish the private transformation, then introduce $\mathcal{L}_{\text{pub}}$ to enforce collapse outside the secret key. Once both behaviors are satisfied, we optionally add the adversarial objective $\mathcal{L}_{\text{adv}}$, followed by the robustness training against key-recovery/aliasing. Stage progression occurs only after all objectives up to the current stage are met.

6.4 Evaluations on Common Benchmarks

Attackers may obtain white-box access to the defense, **excluding $\boldsymbol{z}^\star$ and π**. They can hence optimize AEs in public mode ($\boldsymbol{z}\neq\boldsymbol{z}^\star$) by directly backpropagating gradients through the composed pipeline, while the owner evaluates them in private mode (Fig. 6.1). In

Chapter 5, approaches that simply extend common attacks by additionally differentiating through the defense were referred to as *standard adaptive attacks* (see **Definition** 5.2.2), as this constitutes the baseline methodology for evaluating preprocessing defenses, while attacks crafted solely against the classifier and evaluated post hoc on the composed system often lead to inflated robustness estimations [34]. There, this method was shown sufficient to expose the vulnerabilities of *DBP*. In contrast, *KEYSHIFT* separates public and private decision functions, with the explicit objective of making successful exploitation contingent on recovering hidden information unavailable to the attacker.

Nonetheless, since targeting public mode is the strongest white-box attack possible without knowledge of the hidden space (see §6.5) and is naturally realized through standard gradient-based optimization, we compare *KEYSHIFT*'s robustness under this protocol on common benchmark adversarial attacks against baselines. If we can show that the private behavior is non-extractable, which is precisely the objective of the *advanced adaptive attack* evaluations reported in §6.5, then these results become meaningful as they bound the risk posed by real-world adversaries [34]. Otherwise, attackers could optimize *AE*s directly against the private decision boundaries, breaking the security premise.

6.4.1 Setup

Datasets. We evaluate on ***CIFAR-10*** [139], ***CIFAR-100*** [139], and ***YouTube-Faces*** [264]. ***CIFAR-10/100*** are standard robustness benchmark datasets enabling direct comparison with prior work [14, 257, 207], while ***YouTube-Faces*** contains 1,283 identities at 224×224 resolution, providing a more realistic large-scale setting and allowing us to evaluate the scalability of our approach.

Architectures & Training. Our defense places a preprocessor P in front of a pretrained classifier, making it classifier-agnostic. We hence use common classifier architectures: a WRN-28-10 (*RobustBench*) [54] for ***CIFAR-10***, a ResNet-101 for ***CIFAR-10*** [136], and a ResNet-50 (trained by us) for ***YouTube-Faces***; all classifiers are non-adversarially trained, to isolate *KEYSHIFT*'s robustness. P is a U-Net extended with ConvolveCond (see §6.3.2). We train P using the curriculum from §6.3.2. We use $\sigma = 50.0$ for *KEYSHIFT*'s key space to enforce high entropy (experiments with lower values are in §6.5). Further details are in Appendix D.2.

Adversarial Attacks. We use *AutoAttack* (*AA*) [53] and *PGD* [141] as common benchmarks with $\epsilon_\infty = 8/255$ and $\epsilon_2 = 0.5$ for ***CIFAR***, and $\epsilon_\infty = 8/255$ for ***YouTube-Faces***.

We also evaluate Perceptual *PGD* [143] and our Low-Frequency (*LF*) attack [131] proposed in §5.5, using default hyperparameters (Appendix D.2). These capture additional realistic threat models and provide a stricter test of defense robustness. All attacks were conducted on a set of $1,024$ samples from each dataset.

Baselines. We include three strong *AT* baselines (trained on ℓ_∞) [207, 14, 257] from *RobustBench* [54], and the diffusion-based defense *DiffPure* [182]. Attacks are run manually against the baselines. Due to the computational cost of *DiffPure*, we evaluate it only under *LF* and *AA*-ℓ_∞ on ***CIFAR-10***, similar to Chapter 5.

6.4.2 Results

Clean Accuracy. For *KEYSHIFT*, we report results under public and private modes. As stated throughout, **robustness is provided only in private mode**; public mode collapses to backbone classifier performance, offering no robustness as attackers can access its decision function (see Table 6.1). In private mode, robustness incurs some clean accuracy degradation, consistent with *AT* trade-offs. Yet, unlike *AT*, which alters classifier weights, *KEYSHIFT* offers flexibility: the system may be used in private mode for security-critical deployments, or in public mode in low-risk environments, both to recover full clean accuracy, and to prevent private-behavior leakage when the task does not allow for operational constrains—see §6.2.1. We thus report public-mode accuracy as a complementary but practical metric.

Note. Public-mode deployment can equivalently bypass P entirely and invoke the classifier alone—yielding *exactly* the reported numbers (which we obtained by invoking P in public mode followed by classification), since P contributes nothing there beyond negligible inference overhead. The clean accuracy is thus not enhanced by training P. Yet, regardless of whether P is involved or the classifier is directly called, both approaches qualify as public-mode *KEYSHIFT* in the low-stakes deployments considered above since the classifier retains its clean accuracy precisely because *KEYSHIFT* permits it by not changing its weights (as opposed to *AT*).

In private mode, P applies a sample-adaptive transformation that induces the predefined key-conditioned label permutation. This increases complexity: P becomes active and must realize a class-conditional, input-dependent mapping without context beyond the input, effectively acting as a classifier. As expected, this reduces clean accuracy relative to public mode—see Table 6.1. For reference, among the considered *AT* methods, Wang et al. [257] and MeanSparse [14] achieve higher clean accuracies but rely on tens of millions of additional

Dataset	Method	Clean	AA-ℓ_∞	AA-ℓ_2	PGD-ℓ_∞	LF	PPGD
CIFAR-10	Rebuffi et al. [207]	89.94	67.48	68.16	70.12	1.07	9.96
	Wang et al. [257]	94.34	72.66	71.88	74.80	0.29	9.57
	MeanSparse [14]	95.31	78.32	77.64	80.57	0.59	16.21
	DiffPure [182]	92.19	39.45	N/A	N/A	0.39	N/A
	KEYSHIFT	**86.04** (94.82)	**65.23** (0.00)	**77.83** (0.10)	**73.83** (0.00)	**68.75** (0.00)	**71.39** (1.07)
CIFAR-100	Rebuffi et al. [207]	63.96	36.43	41.70	39.10	1.37	12.79
	Wang et al. [257]	74.90	43.65	46.88	48.24	0.29	10.16
	MeanSparse [14]	74.90	45.51	50.20	50.00	0.39	14.45
	KEYSHIFT	**62.30** (90.43)	**34.38** (0.00)	**50.68** (0.39)	**42.29** (0.00)	**45.02** (0.00)	**37.30** (0.00)
YT-Faces	*KEYSHIFT*	**93.16** (99.12)	**57.81** (10.94)	**N/A**	**64.84** (12.79)	**81.64** (2.15)	**73.24** (0.59)

Table 6.1: Robust accuracy (%) across datasets. For *KEYSHIFT*, we report both **private-mode** and (public-mode) results.

training samples, while Rebuffi et al. [207] provide a competitive baseline with little extra data and comparable clean performance (63.96% vs. 62.30% on ***CIFAR-100*** and 89.94% vs. 86.04% on ***CIFAR-10***). *DBP* (*DiffPure* [182]) achieves strong clean accuracy on ***CIFAR-10*** but it relies on intensive computations at inference time and pretrained diffusion models, limiting its practicality. Moreover, as we show next, its robustness degrades under correct gradient evaluation, undermining its clean accuracy when examined as a robust defense.

The larger gap on ***CIFAR-100*** is consistent with the known increased difficulty of this dataset (e.g., Rebuffi et al. [207] in Table 6.1) due to its low resolution and limited samples per class, whereas most high-robustness defenses on *RobustBench* [54] leverage massive external data or heavy pretraining to improve (clean and robust) accuracy. We include two such models [257, 14] despite these unfair advantages they enjoy since our focus is robust accuracy: as shown later, their robustness degrades under various attacks, while *KEYSHIFT* maintains robustness.

On ***YouTube-Faces*** (not considered by the baselines), *KEYSHIFT* achieves high clean accuracy. While it involves far more classes than ***CIFAR-100***, it provides higher resolution and significantly more samples per class, enabling clearer separation. ***YouTube-Faces*** shows *KEYSHIFT*'s scalability to high-dimensional inputs and large class counts, and its applicability to real-world authentication tasks that motivate our security-focused deployment model.

Robust Accuracy. *KEYSHIFT* (private mode) exhibits strong robustness under ℓ_∞ attacks—the common threat model used to train the *AT* baselines [207, 257, 14]. Under *AA*-ℓ_∞, it achieves 65.23% and 34.38% accuracy on ***CIFAR-10*** and ***CIFAR-100***—within 2–3% of Rebuffi et al. [207], the no-extra-data baseline, and remaining competitive with large-scale *AT* models [14, 257]. *DiffPure* [182] attains substantially lower robustness under *AA*-ℓ_∞, with *KEYSHIFT* exceeding it by 25.75%. Under *PGD*-ℓ_∞, *KEYSHIFT* attains 73.83% and 42.29%, outperforming Rebuffi et al. [207] on both datasets and trailing the strongest *AT* models by less than 8%. This improvement is consistent with the adversarial objective incorporated in training (see §6.3.2), which trains on samples generated with *PGD*. Under *AA*-ℓ_2, *KEYSHIFT* achieves SOTA robustness on ***CIFAR-10*** and ***CIFAR-100***, slightly surpassing the strongest *AT* baseline [14] but improving upon Rebuffi et al. [207] by nearly 9-10%. **Crucially, under the low-frequency attack (*LF*), *KEYSHIFT* achieves 68.75% and 45.02% on *CIFAR-10* and *CIFAR-100*, while all baselines collapse to near-zero ($\leqslant$1.37%). A similar pattern emerges under the perceptual attack (*PPGD*).** On ***YouTube-Faces***, *KEYSHIFT* similarly maintains strong robustness across attacks ($\geqslant$ 57.81%).

Public-mode accuracy degrades across datasets, confirming attack convergence. Since public and private modes induce different decision boundaries, public *AE*s transfer weakly to private mode, consistent with the intended design. *KEYSHIFT* is fully differentiable, rendering BPDA [20] unnecessary (gradient-based attacks collapse public-mode robustness).

These results indicate that separating private and public decision spaces yields robustness across datasets and structurally distinct threat models, substantially outperforming prior defenses on unseen attacks.

6.5 Evaluations against Advanced Adaptive Attacks

Following [34], we devote substantial experiments to *advanced adaptive adversaries*. For any input, a successful private-mode *AE* may be one that is also effective in public mode or depend on behavior unique to the private decision function. In both cases, public-mode access lacks direct information about the private boundary and thus cannot systematically craft *AE*s against it beyond the public→private transfer attacks considered in §6.4. This motivates dedicated attacks that attempt to infer information about the private behavior. In §6.5.1, we evaluate black-box attacks that may exploit (limited) access to the private-mode (deployment) API for this purpose beyond the white-box access to the public mode defense considered in §6.4. In contrast, §6.5.2 studies more sophisticated extraction attacks that

directly attempt to recover hidden private behavior through optimization-based searches and architectural analysis.

Advanced adaptive evaluations can be computationally intensive, particularly when exploring multiple attack variants or large query budgets. Consequently, some experiments focus on ***CIFAR-10*** as a representative benchmark, while additional datasets are included where necessary to verify that the observed behavior generalizes.

6.5.1 Black-box Attacks

Black-box attacks are typically weaker than white-box attacks since they lack direct access to the defense's parameters and gradients [34]. Thus, the term "adaptive attacks" is usually reserved for stronger defense-aware strategies that exploit such access. *KEYSHIFT*, however, creates a different setting due to its separation between public and private operation modes. While attackers may obtain white-box access to the public-mode system, the feedback it provides is inherently incomplete, as demonstrated in §6.4. Hence, it is possible that the outputs returned by querying the deployed private-mode system through the owner's API may alone expose more useful information regarding the hidden process to enable more powerful attacks. Thus, unlike traditional defenses that operate under a single decision function, black-box interaction in *KEYSHIFT* could provide access to a stronger oracle than the attacker-accessible white-box interface despite the lack of gradients.

We evaluate two common black-box paradigms: *query-based attacks*, which craft an *AE* for a given input by only monitoring how the target system's outputs change during the iterative optimization process to identify useful directions to update the sample, and *surrogate-based attacks*, which query the target system on a set of inputs and use the assigned labels to train substitute models that approximate its functionality. Afterward, the adversary utilizes their white-box access to these surrogates to generate *AE*s that often also compromise the target [196]. However, our threat model assumes practical restrictions typical for security-critical deployments (e.g., banking) where interaction with the private-mode API is operationally constrained (see §6.2.1). In particular, queries are rate-limited and only hard labels are returned. Consequently, black-box attacks must operate within these limitations.

Setup. Although hard-label query-based attacks typically require hundreds to thousands of queries, exceeding realistic limitations, one may argue that more efficient hard-label methods could exist. Thus, to conservatively assess their risk, we evaluate the *Square* attack [18], an alternative query-based method that instead requires output probabilities (confidence scores) beyond the hard labels allowed under our threat model. Such attacks

are generally more query-efficient, and *Square* is among the strongest such attacks under limited budgets, making it a suitable proxy. By doing so, we **explicitly relax the threat model**, yielding a stronger attack interface to err on the conservative side. We also evaluate surrogate-based transfer attacks using *JBDA* [196] as a representative.

Results. Under realistic budgets (e.g., 20 queries), *KEYSHIFT* maintains high private-mode accuracy (73.63% on ***CIFAR-10***, 40.92% on ***CIFAR-100***) against *Square*. While increasing the query budget indefinitely continues to degrade robustness, attacks utilizing such large budgets do not constitute successful exploitations as they violate the threat model. *JBDA* also fails to compromise private-mode robustness. Full details are in Appendix D.3.

6.5.2 Extraction Attacks

We now evaluate stronger adaptive attacks that explicitly attempt to recover the private behavior. In *KEYSHIFT*, private behavior corresponds to the process in which the pre-processor induces the label permutation π. Thus, for an attacker who does not know the key, extraction amounts to identifying a mechanism that nevertheless causes the defended system to realize this hidden process. Successful extraction would allow attackers to locally "simulate" the private-mode classification logic and craft *AE*s directly against its decision boundaries. Since the deployed system employs the same hidden process, such *AE*s would naturally compromise it as well, violating the core security premise of *KEYSHIFT*.

Key Recovery Attacks

A natural extraction approach is to optimize a candidate key $\boldsymbol{z}$ to activate private behavior and a label remapping $\boldsymbol{\Pi}$ to align private-mode outputs with ground-truth labels (see §6.3.2).

Attack Formulation. The attacker optimizes the objective introduced in §6.3.2:

$$\boldsymbol{z}', \boldsymbol{\Pi}' = \arg\min_{\boldsymbol{z},\boldsymbol{\Pi}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\ell(\boldsymbol{\Pi}\cdot\mathcal{M}(P(\boldsymbol{x},\boldsymbol{z})),y) - \lambda\|P(\boldsymbol{x},\boldsymbol{z})-\boldsymbol{x}\|_2\big]\,, \tag{6.1}$$

subject to $\boldsymbol{\Pi}\in[0,1]^{K\times K}$ and $\mathrm{tr}(\boldsymbol{\Pi}) = 0$.

The matrix $\boldsymbol{\Pi}$ provides a continuous relaxation of the permutation that aligns predictions with ground-truth labels, with the trace constraint preventing collapse to the trivial identity. The regularization term encourages candidate keys that activate private-mode behavior rather than public-mode identity mappings. **A successful recovery must therefore produce a *functional alias*: $(\boldsymbol{z}', \boldsymbol{\Pi}')$ (ideally $\boldsymbol{z}' = \boldsymbol{z}^\star$) for which $\boldsymbol{\Pi}'\mathcal{M}(P(\boldsymbol{x},\boldsymbol{z}'))$**

achieves high classification accuracy on labeled data. Since the identity mapping is excluded, successful optimization can only occur by converging toward keys that activate the private behavior itself. In particular, the preprocessor is explicitly trained to implement the hidden permutation π under valid private-mode keys, making this behavior the only non-trivial high-accuracy solution encoded in the system. Consequently, if optimization were capable of recovering a functional pair $(\boldsymbol{z}', \boldsymbol{\Pi}')$ that maximizes accuracy, it would necessarily find keys that induce predictions consistent with π, with $\boldsymbol{\Pi}'$ simultaneously aligning the resulting outputs back to the ground-truth labels.

While the adversary's goal is fixed—identifying a key that activates private behavior—the primary modeling choice concerns the parameterization of the permutation π. We represent π as a continuous matrix $\boldsymbol{\Pi}$ updated via gradient descent with projection, as it yields stable optimization; alternative relaxations (Sinkhorn-style) led to gradient attenuation and did not converge when tested against previous backdoor-based defenses (i.e., *AI-Guardian* considered below), whereas this formulation does. This motivates our focus on this attack as the strongest high-impact strategy we identified.

Experimental Setup. We first conduct a large-scale study on ***CIFAR-10***, followed by additional results on the other datasets below. The attack assumes a strong white-box adversary with access to the architecture, parameters, and a dataset on which the *KEYSHIFT*-defended classifier performs well in private mode, allowing the attack objectives from *eq.* (6.1) to be optimized. Thus, to evaluate the strongest adversary, we use the preprocessor's (and classifier's) training set (results are reported on the held-out test set after optimization terminates). To rule out insufficient budgets, each recovery attempt runs for 50 epochs (each processing the full training set). Although optimization typically stagnates early, the full duration is allowed. We use batch size 256 and compute expected gradients over the batch when updating both the key candidate and permutation matrix (when applicable, as not all attacks optimize this matrix—see below). The key (and permutation) candidates are independently initialized randomly in each experiment and the dataset is shuffled randomly as well.

We sweep optimization strategies using AdamW with learning rates from 10^{-3} to 10^{2} (larger values lead to numerical instability, while smaller ones cause the attack to stagnate) and instantiating ℓ in *eq.* (6.1) as both the cross-entropy and margin losses. **We consider two recovery formulations**: **(i)** a *joint recovery* attack optimizing a candidate key $\boldsymbol{z}'$ and permutation matrix $\boldsymbol{\Pi}'$, and **(ii)** a *key-only* attack optimizing $\boldsymbol{z}'$ to maximize the classification loss w.r.t. the ground-truth labels when samples are propagated through the preprocessor under this key—since all non-secret keys activate public behavior, this loss should remain small; thus keys that inflate the loss may correspond to aliases of the secret key. For all combinations of these strategies we run experiments both with and without

the regularization term in *eq.* (6.1). Additionally, we repeat the joint recovery attack (formulation (i) with the regularization term) using SGD while sweeping six logarithmically spaced learning rates between 10^{-3} and 10^{2}. To rule out gradient masking, we also perform gradient-free optimization using *SPSA* [232], estimating gradients from 256 stochastic perturbations per step and updating the key candidate with AdamW (six experiments with learning rates from 10^{-3} to 10^{2}). For *SPSA*, we use $\delta = 0.01$; larger or smaller values were verified not to improve performance in control trials. Whenever the permutation matrix is optimized, it is updated using Adam with learning rate 0.05 (varying this rate provided no advantage).

Note that this attack is strictly stronger than the one used during robustness training (see §6.3.2), where candidate keys are searched using only 100 AdamW optimization steps with random learning rates between 0.1 and 10^{2} sampled at each training step to encourage complete key isolation. In contrast, the evaluation here explores substantially larger optimization budgets and a broader set of loss objectives, optimizers, and recovery strategies, including gradient-free methods. Robustness therefore stems from isolation of the private key region (§6.3.2), rather than overfitting to a specific attack.

Known-Permutation Recovery Attack: As discussed earlier, the main modeling choice in the extraction attack concerns how to represent the permutation π. When π is known, the attacker's objective simplifies to finding a key candidate $\boldsymbol{z}'$ s.t. $\mathcal{M}(P(\boldsymbol{x}, \boldsymbol{z}'))$ achieves high accuracy under labels permuted by π. This reduces the problem to a standard optimization-based search over the key space. To rule out the possibility that the failure of the previous attack stems from an imperfect representation of π, we evaluate a strictly stronger variant in which π is known. This removes the permutation-search component and thus provides an upper bound on recovery performance. Recall that *KEYSHIFT* achieves robustness by isolating the secret region. Consequently, revealing π should not expose useful optimization signal for recovering a valid key alias. This experiment verifies that property. As before, we optimize the key candidate using AdamW with learning rates from 10^{-3} to 10^{2}. We run six experiments across this range, each for 10 epochs over the entire dataset. In practice, optimization stagnates early and fails to make progress.

Results. Across all optimizer, loss, and regularization combinations, the recovered keys yield private-mode accuracies between 0.45% and 0.63% on ***CIFAR-10***, far below the 10% chance level. At the same time, public-mode accuracy remains above 94%, indicating that optimization consistently leaves the preprocessor in its public identity regime rather than activating private behavior. When permutation matrices are optimized jointly with the key, the inferred mappings $\arg\max \boldsymbol{\Pi}'$ exhibit negligible agreement with the true permutation π. These results hold even when the permutation is known and when gradients are replaced by

SPSA-based estimates, demonstrating that the recovery objective provides no exploitable signal for discovering keys that activate private behavior.

Additional Validation. We perform two additional validations to verify that the recovery procedure is effective whenever exploitable signal exists and to contrast the behavior of *KEYSHIFT* with prior defenses:

Low-Entropy KEYSHIFT Baseline: **To further validate the importance of the high-entropy key space emphasized throughout this chapter**, we train a baseline *KEYSHIFT* preprocessor on ***CIFAR-10*** using a low-entropy key distribution ($\sigma = 1.0$) while keeping the same backbone classifier used in our main experiments. Training converges normally, including the robustness stage in which key-recovery attacks are executed for 100 optimization steps. However, during evaluation with larger optimization budgets, the key is consistently recovered (e.g., under formulation (i) using AdamW with learning rate 0.01 running for several epochs on the entire training dataset). This occurs despite the model being trained to resist similar attacks, indicating that robustness training alone is insufficient when the key space has low entropy. As optimization step budgets increase, attacks can explore larger regions of the key space and eventually recover functional aliases of the secret key. Once the key and permutation are recovered, *AE*s crafted under the recovered configuration successfully defeat the protected classifier.

Attack on AI-Guardian: Like *KEYSHIFT*, *AI-Guardian* [302] deploys a hidden decision mode in which classifier outputs are permuted to conceal the true decision boundaries. However, unlike *KEYSHIFT*, it does not construct a high-entropy key space in which this behavior can be hidden. Instead, private-mode evaluation relies on a backdoor trigger: when an input $\boldsymbol{x}$ is augmented with a trigger defined by mask $\boldsymbol{m}$ and pattern $\boldsymbol{p}$, the classifier—trained on both clean and backdoored inputs—maps samples from each source class to a target class defined by a hidden permutation.

Our recovery attack can be adapted to this setting by replacing the key candidate in *eq.* (6.1) with learnable backdoor parameters $(\boldsymbol{m}, \boldsymbol{p})$. The attacker jointly optimizes a permutation matrix $\boldsymbol{\Pi}$ and trigger parameters $(\boldsymbol{m}, \boldsymbol{p})$ so that predictions on triggered inputs become consistent with $\boldsymbol{\Pi}$ across all classes. Unlike the previously reported break of *AI-Guardian* [29], which requires confidence score access from the private-mode deployment, this attack operates entirely offline and requires no interaction with the deployment environment, which may not be feasible (e.g., under the security-critical deployment constraints we assume, scores are not returned).

Applying this formulation to a ***CIFAR-10*** classifier trained with *AI-Guardian*, the attack reliably recovers the hidden permutation and a near-perfect trigger pattern. Using the recovered trigger and permutation, we generate *AE*s with *PGD* against the triggered

classifier and evaluate them under the true deployment configuration. This reduces the classifier accuracy from 85.94% on clean inputs to 19.43%, demonstrating that the recovery attack is highly effective when exploitable signal is present. This behavior is consistent with the low-entropy private space induced by such backdoor-based constructions—similar to our σ=1 baseline, where even robustness training fails to prevent recovery under increased optimization steps.

These results confirm the potency of our attack and that high-entropy key spaces are essential for preventing recovery.

CIFAR-100 & YouTube-Faces. We also evaluate recovery on the ***YouTube-Faces*** defended system using AdamW with learning rates between 10^{-3} and 10^{2} under the original objective (i). Across configurations, the recovered key candidates fail to activate the private decision mode. Private-mode accuracy is near chance (0.0%–0.21%) with permutation recovery accuracy $\leqslant 0.39\%$.

We observe the same behavior on ***CIFAR-100*** under the same attack protocol (AdamW with the permutation-learning objective and regularization) and learning rates. Recovered key candidates cannot induce private-mode behavior, with accuracy remaining at chance level. Permutation recovery accuracy is likewise negligible. For this dataset, we also consider the *SPSA* gradient-free variant with the same objective and observe similar results.

Summary. Across extensive evaluations, key-recovery attacks fail to discover parameters that activate private-mode behavior in *KEYSHIFT*. This holds for multiple optimizers, losses, optimization budgets, gradient-free variants, and even under a stronger oracle attack where the permutation is known. In contrast, the same framework successfully breaks prior defenses and low-entropy baselines, confirming that the attack itself is effective whenever exploitable signal exists. These results demonstrate that robustness arises from isolating the private key region rather than from overfitting to specific attack instances.

Dormant-Neuron Analysis

KEYSHIFT bears a conceptual resemblance to backdoor attacks, where model training is poisoned so that special triggers injected into the inputs activate an alternative behavior. In our case, the auxiliary key $\boldsymbol{z}$ causes the preprocessor to operate in private mode. Unlike classical backdoors, however, our keys are drawn from a high-entropy space and our architecture deliberately entangles the key and input paths to discourage private behavior from being localized to small neuron subsets—see §6.3.2. We evaluate whether such hidden behavior could nevertheless be exposed using backdoor analysis techniques.

Setup. We test this using BAN [276], a SOTA method that perturbs internal neurons to maximize loss on benign inputs. In backdoored models, small but structured neuron perturbations were shown to activate specialized dormant neurons associated with the backdoor, thereby misclassifying benign inputs according to the backdoor mapping [276]. We mount BAN under *public mode* (analogous to "benign" inputs in backdoored models) by fixing a random non-secret $\boldsymbol{z} \neq \boldsymbol{z}^\star$, maximizing classification loss on the dataset w.r.t. the ground-truth labels via noise introduced to the preprocessor's weights. We evaluate on ***CIFAR-10***, varying the noise budget from 0.3 (BAN default) to 30.0.

Noise Budget	$\mathbf{Acc_{pub}}$	$\mathbf{Acc_{priv}}$
0.03	0.9439	0.0047
0.06	0.9441	0.0046
0.066	0.9442	0.0050
0.067	0.1021	0.1006
0.068	0.0977	0.1035
0.3	0.0991	0.0991
3.0	0.1034	0.1018
30.0	0.1006	0.1035

Table 6.2: BAN's performance. Acc_{pub} is classification accuracy under public labeling (ground-truth), while Acc_{priv} is the accuracy under π. Successful recovery would decrease Acc_{pub} while increasing Acc_{priv}.

Results. Table 6.2 reports the evaluation results under various neural noise budgets. We report two metrics: Acc_{pub}, measuring classification accuracy w.r.t. the ground-truth labels (initially high since the model is run in public mode), and Acc_{priv}, measuring agreement with the hidden label permutation. If BAN's optimized noise successfully exposed private behavior, Acc_{pub} would decrease while Acc_{priv} would increase.

We first replicate the BAN noise regime used in the original work [276], sweeping perturbation budgets from 0.3 to 30. For large budgets ($\geqslant 3.0$), Acc_{pub} is already low at the beginning of the optimization, indicating that such noise levels immediately corrupt the model's weights and render the network dysfunctional. At smaller budgets (e.g., the default $\epsilon = 0.3$ used in [276] or below), Acc_{pub} initially remains near 100%, but rapidly collapses to near-random predictions as the optimization progresses—see Table 6.2. In all cases, Acc_{priv} accuracy remains near chance ($\approx 10\%$), indicating that the optimized perturbations do not reveal the hidden mapping but instead destabilize the network's computation.

To determine whether smaller perturbations might activate private behavior without corrupting the model, we perform a binary search to identify the smallest noise magnitude that affects predictions. Acc_{pub} remains near 100% at $\epsilon = 0.03$, while a sharp transition occurs around $\epsilon \approx 0.067$, where predictions collapse to near-random outputs. The three central rows in Table 6.2 $(0.066, 0.067, 0.068)$ zoom into this transition region. Even at the breaking point (0.067), Acc_{priv} remains near chance, confirming that even carefully calibrated neuron perturbations fail to activate private behavior.

Summary. BAN fails to reveal private behavior. This supports the efficacy of the **entanglement** enforced by ConvolveCond (see §6.3.2), which mixes the input $\boldsymbol{x}$ and conditioning $\boldsymbol{z}$ across layers, limiting the separation between benign and backdoor neurons assumed by backdoor analysis methods. Private behavior therefore becomes more rooted in coordinated activation patterns distributed throughout the network rather than a small set of dormant units.

Universal Adversarial Perturbations (*UAP*s)

We finally evaluate a permutation recovery attack based on *UAP*s originally proposed by Carlini et al. [29] in their analysis of *AI-Guardian* [302]. The attack exploits the observation that traditional hidden-behavior defenses (e.g., backdoor-based ones) effectively cause the classifier to associate each input with two labels—the ground-truth label and the hidden mapping. Consequently, when optimizing a universal perturbation $\boldsymbol{\delta}$ to induce misclassification for all samples belonging to a given source class, gradient descent tends to move toward the hidden mapping due to this pre-existing bias, allowing the attacker to infer the concealed permutation from the easiest class translations under **small** universal input perturbations (larger perturbations would overwrite the inputs entirely, causing them to be classified as any target class).

In *KEYSHIFT*, however, the classifier itself contains no hidden behavior: the permutation is implemented entirely within the key-conditioned preprocessor while the classifier remains fixed. Because the key space has high entropy, randomly sampled keys used to invoke the system in public mode remain far from the secret key, causing the preprocessor to collapse to the identity transformation. Thus, *UAP* optimization, which applies small perturbations to the inputs $\boldsymbol{x}$ without access to the secret key, receives gradients associated only with public-mode behavior and therefore observes no direct signal about the hidden permutation, as it fails to alter the preprocessor's behavior (which depends on the distant secret key). In other words, the entropy of the key space isolates exploitable private-mode biases. To validate this claim, we run this attack against *KEYSHIFT*.

Note. While knowledge of the hidden label permutation alone does not directly imply that attackers would be able to recover the private behavior, as demonstrated in our recovery attack experiments above, constructing *UAP*s that successfully reveal the permutation could still compromise the hidden decision process. In particular, optimizing these *UAP*s implicitly searches for input-space directions correlated with private-mode activation, which is governed by the secret key. Thus, if such perturbations existed reliably, they could potentially expose structural cues that enable more advanced (key) extraction strategies than those considered thus far in this section. Therefore, the goal of this experiment is not

merely to evaluate permutation recovery itself, but to determine whether optimization can identify exploitable signals associated with the private mode.

Experimental Setup. Following Carlini et al. [29], we use an SGD optimizer and restrict the optimization budget (learning rate and number of steps) to prevent the perturbation from fully rewriting the inputs. Specifically, we optimize each *UAP* with a learning rate of 0.01, varying the number of steps from 20 to 100 on ***CIFAR-10***'s test-set samples. We run the attack against the *KEYSHIFT*-protected classifier (in public mode), iterating over all source–target class pairs and attempting to find *UAP*s that change the classifier's prediction on these samples from the source to the target. Similar to [29], after each *UAP* optimization we take the majority label over a batch of samples (64 samples in our case) to determine the predicted class under the crafted perturbation.

Results. Table 6.3 summarizes permutation recovery across optimization budgets. For each target label we optimize a universal perturbation and measure whether the corresponding source class under the hidden permutation can be uniquely identified. A recovery is counted as successful only if the perturbation maps the correct source class's majority samples to its target under the hidden permutation while no other class exhibits the same translation. The first column reports the optimization budget. The second column reports how often the perturbation correctly maps the true source class to the target label, regardless of whether the mapping is unique. The third column reports the average number of additional classes whose samples are also mapped to the same target label. The final column reports the number of successful recoveries (unique source identification) across the ten target labels.

For small optimization budgets the perturbations rarely induce consistent class translations. As the optimization budget increases, the true source class is increasingly mapped to the target; however, the perturbations simultaneously map many other classes to the same label. This prevents unique source identification and thus prevents permutation recovery. Across all optimization budgets the attack therefore fails to recover the hidden permutation. The only exception occurs at the 30-step budget, where a single source is successfully recovered; however,

Steps	True Sources Identified	Avg. Other Sources Mapped	Successful Recoveries
20	0/10	0.5	0/10
30	1/10	1.1	1/10
40	2/10	2.7	0/10
50	4/10	4.5	0/10
100	10/10	7.4	0/10

Table 6.3: Permutation recovery using universal adversarial perturbations (*UAP*s). A recovery is successful only if the true source class uniquely maps to the target. Across all step budgets the attack never recovers the hidden permutation.

this case does not generalize to other targets and disappears as the optimization budget increases, indicating an incidental alignment rather than a systematic recovery signal.

Summary. These results show that *UAP*-based permutation recovery fails against *KEYSHIFT*. This behavior is consistent with our design, where hidden behavior is not embedded in the classifier itself but mediated through the high-entropy key-conditioned preprocessor, leaving no persistent bias that *UAP* optimization can exploit.

Concluding Remarks

The results in this section indicate that *KEYSHIFT* complicates private behavior extraction or approximation, resisting search-based, architectural, and realistic black-box attacks, and preserving adversarial robustness as long as the key remains undisclosed, marking a promising direction for future defenses based on this attack-aware paradigm.

6.6 Discussion & Limitations

Performance and Scalability. *KEYSHIFT* introduces minimal inference overhead. However, training the preprocessor can be computationally demanding. For example, ***YouTube-Faces*** required roughly one week on a single A100 GPU (see Appendix D.4 for complexity details). While datasets such as ***ImageNet*** are left to future work due to time constraints, ***YouTube-Faces*** exhibits comparable dimensions and number of classes, and its training size is substantial. Moreover, face-recognition datasets are better aligned with the security-critical deployments we target (e.g., authentication). ***ImageNet*** also contains more training samples and higher task complexity, which would likely require additional compute. Yet, in practice, organizations deploying *KEYSHIFT* can parallelize training across advanced hardware, making it highly feasible. For instance, recent accelerators such as NVIDIA Blackwell systems can offer substantial speedups over A100-class GPUs on large-scale training workloads.

Future Directions. This work serves as a proof of concept. Accuracy could be improved using stronger backbone classifiers, enhanced training regimes, or data augmentation, similar to advances made in *AT* over the years. Furthermore, combining *KEYSHIFT* with adversarially trained classifiers could reduce transferability of public-mode *AE*s, providing an additional layer of protection. We emphasize, however, an honest limitation: in its current form, *KEYSHIFT*'s attained accuracies—in private mode, even absent attacks—do

not yet qualify it to reliably serve the highest-stakes deployments it ultimately targets. We do not claim otherwise. What it does establish is a substantial advance over the state of the art, which frequently collapses entirely under various attack families and adaptive adversaries, whereas *KEYSHIFT* retains significant robustness. Closing the remaining accuracy gap is precisely the direction this work aims to open rather than to have already completed.

Likewise, while we strive to map the private-behavior leakage channels comprehensively, we cannot rule out that future analysis uncovers new ones. This, however, is where the *assumption stress-testing* methodology is most valuable: a newly identified channel need not break the design but can be incorporated as an additional objective and defended against in turn, incrementally hardening the mechanism. Crucially, this is not the familiar cycle of patching a defense after each break: that cycle fails because each fix individually targets a specific demonstrated breach, leaving the true attack surface unbounded. *KEYSHIFT* instead establishes its baseline under worst-case conditions and stress-tests the single assumption (non-extractability) on which its security rests. Within this discipline, closing a newly identified channel need not be a patch that only temporarily restores a broken claim, but a further extension to a systematic hardening procedure already operating under the strongest attacker model, eventually facilitating convergence toward a truly robust system, should one exist. *KEYSHIFT* is thus not offered as a foolproof solution, but as a defensible, extensible construction—secured through transparency and adaptive scrutiny from the outset. This distinction allows it to be strengthened in a principled manner as understanding deepens.

6.7 Conclusion

We introduced *KEYSHIFT*, an adversarial defense that separates attacker-observable computation from private model behavior via key-conditioned preprocessing, achieving strong performance across threat models that degrade existing defenses. *KEYSHIFT* isolates the private decision function in a high-entropy space, making its recovery information-limited. Extensive adaptive evaluation fails to extract the hidden behavior or compromise robustness. Our results suggest that carefully designed mechanisms can remain robust under rigorous testing, offering a promising direction for protecting *ML* systems deployed in high-stakes settings.

Chapter 7

Conclusion & Future Directions

Advances in *ML* are only made possible because we continue to develop theoretical tools and concepts that allow us to tackle real-world problems. However, these concepts are often abstract and do not directly fit the application we seek. Worse yet, they are developed under idealistic conditions that do not hold in reality. Thus, to bridge this gap, we must make assumptions—about the data distribution, the environment, or the threat model. The central premise of this thesis is that these assumptions inevitably become the weakest link as they connect practical systems to their fundamental principles. Consequently, if these assumptions fail, mechanisms based on them may collapse in ways not captured by the theory motivating them. Accordingly, the thesis examined the security of *ML* systems through the lens of their core assumptions across multiple domains, demonstrating how critical failures do not stem from isolated weaknesses of replaceable individual designs, but reflect a broader challenge in how robustness is conceptualized, evaluated, and ultimately designed in practice. The central question investigated was:

> *How can the evaluation and design of ML security mechanisms account for the assumptions underlying their robustness and their implications in order to provide meaningful real-world security guarantees?*

To address this question, this thesis introduced the perspective of *assumption stress-testing*, which systematically surfaces assumptions and constructs adversarial strategies that explicitly target them. Below, we summarize the contributions of this work and outline future research directions.

7.1 Summary of Contributions

Chapters 3 and 4 investigated whether restricting model access suffices to provide a meaningful basis for security, an assumption motivating various high-stakes *ML* applications in real-world use. Chapter 3 subjected deepfake speech detectors to the *assumption stress-testing* framework in this setting through an in-depth analysis of how they distinguish human from synthetic speech, revealing that, despite operating in complex latent spaces, these systems inevitably rely on human-understandable low-level generative artifacts shared across fake audio inputs. Exploiting this insight, the chapter developed specialized temporal and spectral transformations that suppress these cues directly rather than relying on model feedback. The resulting contribution is a fully black-box, efficient attack that enables fake speech to bypass the full voice authentication stack as deployed in high-stakes environments (e.g., banks)—with up to 99% success within only a few authentication attempts, compromising both academic and commercial systems.

Chapter 4 revisited a similar assumption in the context of AI-generated image watermarking, where security relies on the secrecy of the watermark and restricted access to its detection logic. By analyzing how watermarking schemes encode information, *UnMarker* identifies shared spectral structures in which they must embed their watermark signals across images. This reframes watermark removal as a visually constrained signal-level disruption problem, where perturbations interfere with these broader structures, rather than localizing the specific watermark. This chapter contributes a principled analysis of the limitations of modern watermarking solutions against deepfakes and the first attack that is universally effective across watermarking schemes without requiring knowledge of the watermark or detector feedback, compromising seven state-of-the-art schemes, including semantic watermarking previously regarded as a promising direction for trustworthy deepfake detection, reducing the best detection rate to 43%. Together, these chapters showed that obscuring model internals alone does not *necessarily* prevent attackers from identifying and exploiting structural properties inherent to the data itself.

Chapter 5 investigated whether strong theoretical motivation necessarily translates into practical robustness by revisiting diffusion-based purification (*DBP*), a prominent defense class against adversarial examples grounded in the premise that diffusion models can project adversarial inputs back onto the natural data manifold, eliminating malicious perturbations. Through formal analysis and rigorous testing under proper differentiation procedures and advanced adaptive optimization strategies that craft low-frequency adversarial perturbations to survive *DBP*'s stochasticity, the chapter demonstrated that prior robustness claims were substantially inflated due to flawed gradient propagation, suboptimal attack methodologies, and evaluation artifacts—degrading *DBP*'s robustness to near 0%. More broadly, these

findings demonstrated that theoretical justification alone does not guarantee meaningful robustness once practical deployment constraints and adaptive adversarial reasoning are taken into account.

Chapter 6 explored the constructive application of *assumption stress-testing*, motivated by the impracticality of certifiable defenses and the repeated collapse of empirical methods under adaptive evaluation. It introduced *KEYSHIFT*, a key-conditioned defense against adversarial examples that separates attacker-observable behavior from a private deployment-time decision process, making successful exploitation contingent on recovering concealed functionality. Unlike prior approaches relying on hidden behavior, which attackers were able to recover thereby nullifying their security, *KEYSHIFT* adopted assumption-driven reasoning as a design principle, systematically examining its attack surface and striving to construct the private behavior so that recovery becomes an information-limited search problem. Additionally, in contrast to the systems studied in Chapters 3 and 4, *KEYSHIFT* does not rely on hiding model internals or architectures. Instead, following Kerckhoffs' principle, it assumes white-box knowledge of the system while only leveraging operational constraints commonly present in security-critical deployments. Extensive evaluations showed that, without prior knowledge of the embedded behavior, attacker-crafted adversarial examples fail to compromise deployment-time inference, improving robustness where existing defenses break, while various advanced recovery attempts fail to expose this private mechanism. As formally-certified robustness mechanisms remain of little practical applicability, this chapter illustrated how *assumption stress-testing* can serve not only as an evaluative tool, but also to potentially guide the development of empirical defenses whose security remains meaningful under active analysis, establishing a defensible baseline via rigorous adaptive scrutiny from the outset.

A unifying framework for evaluating *ML* security. Beyond the individual attack and defense contributions, this thesis employed them as case studies to advance *assumption stress-testing* as a general framework for systematically analyzing and designing secure *ML* systems. Rather than taking the standard approach of directly evaluating individual systems, which rarely provides insights beyond isolated vulnerabilities of specific designs, this perspective explicitly elevates assumptions into the primary objects of analysis, **demonstrating how violating them leads to catastrophic failures at the conceptual level, compromising the applicability of the concept itself rather than merely a particular implementation.** While assumption-aware reasoning is conceptually a natural approach for evaluating robustness, the central challenge is that systematically applying it is highly nontrivial. Assumption violations rarely correspond directly to tractable optimization objectives, complicating the task of constructing viable strategies to validate them. This often leads to de-prioritizing assumption-level analysis, allowing vulnerabilities such as

those this work uncovered to remain undetected.

Across chapters, the thesis instantiated this *assumption stress-testing* framework by identifying assumptions behind robustness claims, reasoning about how they may fail, and translating these failures into practical attacks. Although it does not provide a universal procedure for automatically stress-testing the assumptions behind arbitrary systems, which is naturally unique to the application, the presented projects developed diverse methodologies for operationalizing their violations across domains. These include universal signal transformations for suppressing generative artifacts in synthetic speech, spectral surrogate objectives for disrupting watermark carriers, structured low-frequency optimization strategies for defeating semantic watermarking and stochastic purification defenses, and adaptive search procedures for evaluating and hardening *KEYSHIFT*. Critically, these methodologies are not confined to the projects that introduced them. The perceptually-constrained low-frequency optimization developed to disrupt watermark carriers (Chapter 4) is the same mechanism that defeats stochastic purification (Chapter 5)—one operationalization principle spanning two structurally distinct security concepts. The framework's reach extends even across modalities: the frequency-level analysis that helped expose the vulnerability of voice authentication (Chapter 3) motivated our spectral optimizations later developed against image watermarking (Chapter 4). Such reuse—of machinery in the first case and of analytical approach in the second—is what distinguishes a framework from a collection of independent attacks: both the reasoning style and the specific techniques transfer across domains, tasks, and assumptions. Therefore, the systematization of *assumption stress-testing* and the accompanying projects serve two primary purposes: i) establishing assumption-aware reasoning as the default lens through which robustness should be evaluated despite its potential difficulty, and ii) providing concrete demonstrations of the analytical and creative process required to carry out such intricate analyses, alongside extensible tools and techniques, serving as a foundational framework for future work to adapt and generalize to various other domains.

7.2 Future Directions

This thesis has shown that many existing systems fail because they rely on assumptions that do not hold under careful adaptive scrutiny. However, the ultimate goal of security research is not merely to invalidate vulnerable mechanisms, but to use rigorous evaluation frameworks such as *assumption stress-testing* to guide the development of trustworthy countermeasures. Moving forward, this motivates a shift toward assumption-aware design paradigms in which adaptive robustness is treated as a baseline requirement that new

mechanisms must consider from the outset, rather than an evaluation process applied retroactively once systems are already widely trusted.

Chapter 6 readily demonstrates one such application in the context of defenses against adversarial examples through *KEYSHIFT*. Yet, under this perspective, many existing and emerging *ML* defenses should be revisited, including those studied throughout this thesis, with our gathered insights serving as foundational building blocks for future designs. As a concrete instance, the analysis in Chapter 4 suggests a direct path toward resolving the open question of defensive watermarking's viability. Rather than continuing to propose schemes that are then broken in turn, one can incorporate robustness to spectrally destructive attacks such as *UnMarker* as an explicit training objective, hardening a watermark to its operational limit and thereby *gathering defensible evidence* for non-removability, rather than relying on fragile assumptions (see §4.7.3). This re-frames our destructive attack as an instrument of constructive design, exemplifying the assumption-aware paradigm advocated above.

At the same time, many rapidly emerging domains require similarly rigorous treatment. In particular, extending *assumption stress-testing* to large language models (*LLMs*) and agentic systems represents a critical research direction. As these systems become increasingly integrated into high-impact applications, understanding their vulnerability to adversarial attacks such as jailbreaks [259] and prompt injections [93] becomes essential. Yet, unlike perceptual domains such as vision and audio, attacks against *LLMs* are harder to formalize and optimize, context-dependent, and unfold over long interactions. Consequently, rigorously evaluating defenses against these threats becomes substantially more difficult as well [206]. Existing approaches either rely on static benchmarks that risk creating a false sense of security, or depend heavily on manually engineered attack strategies and human red-teaming efforts—an approach that is slow and delays rigorous validation of newly proposed defenses. As a result, many mechanisms are initially viewed as secure or promising before their underlying assumptions are carefully scrutinized, only for them to later fail under stronger adaptive analysis [181].

This raises a core challenge: if systematically operationalizing assumption violations is already difficult in structured perceptual domains, how can this process be scaled to *LLMs*, where adversarial reasoning itself often poses additional challenges? The problem is further amplified by the diversity of applications these systems support and by the fundamentally different assumptions underlying rapidly emerging defense proposals. One promising direction is to move beyond the current use of *LLMs* merely as execution engines for predefined human-designed attacks, and instead investigate autonomous agentic systems capable of performing assumption-aware adaptive evaluation themselves. Such systems could analyze defense descriptions, infer their assumptions, and iteratively construct

attack strategies through structured reasoning, execution, and feedback-driven refinement. Pursuing this direction introduces substantial challenges, including coordinating agent behaviors, defining reliable reward functions for open-ended attack synthesis, diagnosing failure sources, and supporting long-horizon adaptive reasoning. Nevertheless, addressing these challenges could provide a scalable alternative to slow manual red-teaming, while also offering broader insight into the extent to which automated systems can approximate human adaptive reasoning.

More broadly, such agentic systems could extend beyond attacking *LLM*s and serve as the first viable path toward automating the most difficult aspect of *assumption stress-testing*: translating abstract assumption failures into concrete adversarial analyses across domains. If successful, this could help close the gap between evaluation and design, enabling the constructive use of *assumption stress-testing*, exemplified by *KEYSHIFT*, where robustness becomes a continuously refined design objective rather than a property validated only after deployment.

Another critical aspect of *ML* is the privacy of its users. Beyond adversarial robustness, privacy-preserving technologies represent a domain where assumption-aware analysis could provide significant benefits. Despite substantial progress, however, many challenges in this area remain unresolved. A prominent mechanism underlying modern privacy-preserving solutions is differential privacy (*DP*) [73], which provides formal guarantees bounding potential privacy leakage when integrated into *ML* algorithms. Yet, these guarantees are often derived under worst-case conditions and may therefore be overly pessimistic. A central question is then whether such bounds are tight in practice, or whether substantially stronger privacy may hold against realistic adversaries unable to instantiate the extreme settings required to attain the theoretical limit. This line of inquiry, commonly referred to as privacy auditing, has recently gained traction through empirical attacks demonstrating that sufficiently powerful adversaries with extensive access to training procedures, gradients, or model architectures can indeed approach the formal privacy bounds [180, 36, 77]. While these results constitute important foundational progress, they still do not fully capture practical threat models, where attackers are often far more constrained. By explicitly formulating assumptions about real-world adversarial capabilities and systematically analyzing how these assumptions influence privacy guarantees, *assumption stress-testing* could enable more practically grounded auditing methodologies and mitigation strategies. Such approaches may ultimately help quantify privacy loss under realistic deployment conditions, sharpening how *DP* should be interpreted and relied upon in practice.

Letter of Copyright Permission

Portions of this thesis are based on material previously published by the author. The reuse of that material in this thesis is governed by the terms set out below.

IEEE-Copyrighted Material. Chapter 4 is based on the following paper, for which copyright was transferred to the Institute of Electrical and Electronics Engineers (IEEE):

- A. Kassis and U. Hengartner, "UnMarker: A Universal Attack on Defensive Image Watermarking," in *Proceedings of the 46th IEEE Symposium on Security and Privacy (S&P)*, 2025.

The IEEE does not require individuals working on a thesis to obtain a formal reuse license. In accordance with the IEEE reuse policy for theses and dissertations, the following copyright and credit notice is provided:

> © 2025 IEEE. Reprinted, with permission, from A. Kassis and U. Hengartner, "UnMarker: A Universal Attack on Defensive Image Watermarking," Proceedings of the 46th IEEE Symposium on Security and Privacy (S&P), May 2025.

The accepted (author) version of this material is reproduced here, as required by the IEEE. The IEEE-published version is available via IEEE Xplore: https://ieeexplore.ieee.org/document/11023303.

In reference to IEEE-copyrighted material, the IEEE does not endorse any of the University of Waterloo's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE-copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a license from RightsLink.

Author-Retained Material. For the following works, the authors retain copyright, and the material is reproduced and adapted in this thesis by the author on that basis. Attribution to the original publications is provided below.

Chapter 3 is based on a paper published in the IEEE Symposium on Security and Privacy under the IEEE Computer Society's "Delayed-Open Access" model. Under the

terms of that license, IEEE held an exclusive license for a period of twelve (12) months from inclusion in the IEEE Xplore Digital Library, after which the authors retain ownership and IEEE holds only a non-exclusive license. That exclusivity period has expired, and the work is available as open access:

- A. Kassis and U. Hengartner, "Breaking Security-Critical Voice Authentication," in *Proceedings of the 44th IEEE Symposium on Security and Privacy (S&P)*, 2023, published under the IEEE Computer Society Delayed-Open Access model. Copyright retained by the authors. Available: https://ieeexplore.ieee.org/document/10179374

Chapter 5 is based on a paper published at the Conference on Neural Information Processing Systems (NeurIPS) under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license, with copyright retained by the authors:

- A. Kassis, U. Hengartner, and Y. Yu, "DiffBreak: Is Diffusion-Based Purification Robust?," in *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025. Available: https://openreview.net/forum?id=Z7eXOBcxE9

As an author of these works, the material is reproduced and adapted in this thesis in a manner consistent with the terms under which each was published.

References

[1] H. Abdullah, M. Rahman, W. Garcia, K. Warren, A. Swarnim Yadav, T. Shrimpton, and P. Traynor. Hear "No Evil", see "Kenansville": Efficient and transferable black-box attacks on speech recognition and voice identification systems. In *IEEE Symposium on Security and Privacy (SP)*, 2021.

[2] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin R. B. Butler, and Joseph Wilson. Practical hidden voice attacks against speech and speaker recognition systems. In *Network and Distributed System Security Symposium (NDSS)*, 2019.

[3] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. The faults in our ASRs: An overview of attacks against automatic speech recognition and speaker identification systems. In *IEEE Symposium on Security and Privacy (SP)*, 2020.

[4] Aculab. Enghouse Aculab Telco and Cloud Solutions. URL https://www.aculab.com.

[5] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[6] Muhammad Ejaz Ahmed, Il-Youp Kwak, Jun Ho Huh, Iljoo Kim, Taekkyung Oh, and Hyoungshick Kim. VOID: A fast and light voice liveness detection system. In *USENIX Security Symposium (USENIX Security)*, 2020.

[7] Shimaa Ahmed, Yash Wani, Ali Shahin Shamsabadi, Mohammad Yaghini, Ilia Shumailov, Nicolas Papernot, and Kassem Fawaz. Tubes among us: Analog attack on automatic speaker identification. In *USENIX Security Symposium (USENIX Security)*, 2023.

[8] Avinash Ahuja. NVIDIA confidential computing to help expand Apple's private cloud compute. *NVIDIA's Company Blog*, 2026. URL https://blogs.nvidia.com/blog/nvidia-confidential-computing-apple-private-cloud-compute/?ncid=so-face-155679&linkId=100000425748219.

[9] Jamal Kaid Mohammed Ali, Muayad Abdulhalim Ahmad Shamsan, Taha Ahmed Hezam, and Ahmed A. Q. Mohammed. Impact of ChatGPT on learning motivation: Teachers and students' voices. *Journal of English Studies in Arabia Felix*, 2023.

[10] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. Did you hear that? adversarial examples against automatic speech recognition. In *Workshop on Machine Learning for Audio, at NeurIPS*, 2018.

[11] Amazon. Mechanical Turk. URL https://www.mturk.com/.

[12] Amazon. Amazon Connect. URL https://aws.amazon.com/connect.

[13] Amazon. Amazon Connect Voice ID. URL https://aws.amazon.com/connect/voice-id/.

[14] Sajjad Amini, Mohammadreza Teymoorianfard, Shiqing Ma, and Amir Houmansadr. MeanSparse: Post-training robustness enhancement through mean-centered feature sparsification. In *International Conference on Learning Representations (ICLR)*, 2025.

[15] Bang An, Mucong Ding, Tahseen Rabbani, Aakriti Agrawal, Yuancheng Xu, Chenghao Deng, Sicheng Zhu, Abdirisak Mohamed, Yuxin Wen, Tom Goldstein, and Furong Huang. WAVES: Benchmarking the robustness of image watermarks. In *International Conference on Machine Learning (ICML)*, 2024.

[16] S. Abhishek Anand, Jian Liu, Chen Wang, Maliheh Shirvanian, Nitesh Saxena, and Yingying Chen. EchoVib: Exploring voice authentication via unique non-linear vibrations of short replayed speech. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2021.

[17] Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 1982.

[18] 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)*, 2020.

[19] ANZ Bank. Voice ID in the ANZ app. URL https://www.anz.com.au/ways-to-bank/mobile-banking-apps/voice-id/.

[20] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

[21] Mitali Bafna, Jack Murtagh, and Nikhil Vyas. Thwarting adversarial examples: An L0-robust sparse Fourier transform. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[22] Brian R. Bartoldson, James Diffenderfer, Konstantinos Parasyris, and Bhavya Kailkhura. Adversarial robustness limits via scaling-law and human-alignment studies. In *International Conference on Machine Learning (ICML)*, 2024.

[23] Ashley Belanger. OpenAI, Google will watermark AI-generated content to hinder deepfakes, misinfo. *Ars Technica*, 2023. URL https://arstechnica.com/ai/2023/07/openai-google-will-watermark-ai-generated-content-to-hinder-deepfakes-misinfo/.

[24] Ning Bi, Qiyu Sun, Daren Huang, Zhihua Yang, and Jiwu Huang. Robust image watermarking based on multiband wavelets and empirical mode decomposition. *IEEE Transactions on Image Processing*, 2007.

[25] Logan Blue, Kevin Warren, Hadi Abdullah, Cassidy Gibson, Luis Vargas, Jessica O'Dell, Kevin Butler, and Patrick Traynor. Who are you (i really wanna know)? detecting audio deepfakes through vocal tract reconstruction. In *USENIX Security Symposium (USENIX Security)*, 2022.

[26] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian J. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[27] Xiaoyu Cao and Neil Zhenqiang Gong. Understanding the security of deepfake detection. In *International Conference on Digital Forensics and Cyber Crime (ICDF2C)*, 2021.

[28] Nicholas Carlini. A partial break of the honeypots defense to catch adversarial attacks. *arXiv preprint arXiv:2009.10975*, 2020.

[29] Nicholas Carlini. A LLM assisted exploitation of AI-Guardian. *arXiv preprint arXiv:2307.15008*, 2023.

[30] Nicholas Carlini. Cutting through buggy adversarial example defenses: Fixing 1 line of code breaks SABRE. *arXiv preprint arXiv:2405.03672*, 2024.

[31] Nicholas Carlini and David Wagner. MagNet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017.

[32] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

[33] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Security and Privacy Workshops (SPW)*, 2018.

[34] 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 arXiv:1902.06705*, 2019.

[35] Nicholas Carlini, Florian Tramèr, Krishnamurthy Dj Dvijotham, Leslie Rice, Mingjie Sun, and J. Zico Kolter. (Certified!!) adversarial robustness for free! In *International Conference on Learning Representations (ICLR)*, 2023.

[36] Tudor Cebere, Aurélien Bellet, and Nicolas Papernot. Tighter privacy auditing of DP-SGD in the hidden state threat model. In *International Conference on Learning Representations (ICLR)*, 2025.

[37] Guangke Chen, Sen Chen, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. Who is real bob? adversarial attacks on speaker recognition systems. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

[38] Huanran Chen, Yinpeng Dong, Shitong Shao, Zhongkai Hao, Xiao Yang, Hang Su, and Jun Zhu. Your diffusion model is secretly a certifiably robust classifier. In *International Conference on Machine Learning (ICML)*, 2024.

[39] Huanran Chen, Yinpeng Dong, Zhengyi Wang, Xiao Yang, Chengqi Duan, Hang Su, and Jun Zhu. Robust classification via a single diffusion model. In *International Conference on Machine Learning (ICML)*, 2024.

[40] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[41] Xinquan Chen, Xitong Gao, Juanjuan Zhao, Kejiang Ye, and Cheng-Zhong Xu. AdvDiffuser: Natural adversarial example synthesis with diffusion models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[42] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. Devil's Whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices. In *USENIX Security Symposium (USENIX Security)*, 2020.

[43] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Workshop on Deep Learning for Recommender Systems (DLRS), at RecSys*, 2016.

[44] Cinta Infinita. Android: How to bypass root check and certificate pinning. *Medium*, 2020. URL https://medium.com/@cintainfinita/android-how-to-bypass-root-check-and-certificate-pinning-36f74842d3be.

[45] Moustapha M. Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[46] Clipdrop. Stable diffusion API. URL https://clipdrop.co/stable-diffusion.

[47] Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

[48] Ramon Corretge. Praat Vocal Toolkit. URL https://www.praatvocaltoolkit.com.

[49] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[50] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *ACM Conference on Recommender Systems (RecSys)*, 2016.

[51] Joseph Cox. How i broke into a bank account with an AI-generated voice. *VICE*, 2023. URL https://www.vice.com/en/article/dy7axa/how-i-broke-into-a-bank-account-with-an-ai-generated-voice.

[52] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 2018.

[53] 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)*, 2020.

[54] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: A standardized adversarial robustness benchmark. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[55] Dustin T. Crystal, Nicholas G. Cuccolo, Ahmed Ibrahim, Heather Furnas, and Samuel J. Lin. Photographic and video deepfakes have arrived: How machine learning may influence plastic surgery. *Plastic and Reconstructive Surgery*, 2020.

[56] Jiequan Cui, Zhuotao Tian, Zhisheng Zhong, Xiaojuan Qi, Bei Yu, and Hanwang Zhang. Decoupled Kullback-Leibler divergence loss. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[57] Steffen Czolbe, Oswin Krause, Ingemar Cox, and Christian Igel. A loss function for generative neural networks based on Watson's perceptual model. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[58] 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 arXiv:1705.02900*, 2017.

[59] Rohan Kumar Das, Xiaohai Tian, Tomi Kinnunen, and Haizhou Li. The attacker's perspective on automatic speaker verification: An overview. In *Interspeech*, 2020.

[60] Emilia David and Alex Heath. Meta says you better disclose your AI fakes or it might just pull them. *The Verge*, 2024. URL https://www.theverge.com/2024/2/6/24062388/meta-ai-photo-watermark-facebook-instagram-threads.

[61] Adrienne De Ruiter. The distinct wrong of deepfakes. *Philosophy & Technology*, 2021.

[62] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[63] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[64] Hitesh Dharmdasani. Android-Rootkit. URL https://github.com/hiteshd/Android-Rootkit.

[65] Guneet S. Dhillon, Kamyar Azizzadenesheli, Zachary C. Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations (ICLR)*, 2018.

[66] Alec F. Diallo and Paul Patras. SABRE: Cutting through adversarial noise with adaptive spectral filtering and input reconstruction. In *IEEE Symposium on Security and Privacy (SP)*, 2024.

[67] Yi-Yang Ding, Jing-Xuan Zhang, Li-Juan Liu, Yuan Jiang, Yu Hu, and Zhen-Hua Ling. Adversarial post-processing of voice conversion against spoofing detection. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2020.

[68] Yi-Yang Ding, Hao-Jian Lin, Li-Juan Liu, Zhen-Hua Ling, and Yu Hu. Robustness of speech spoofing detectors against adversarial post-processing of voice conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[69] Chengdong Dong, Ajay Kumar, and Eryun Liu. Think twice before detecting GAN-generated fake images from their spectral domain imprints. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[70] Ziping Dong, Chao Shuai, Zhongjie Ba, Peng Cheng, Zhan Qin, Qinglong Wang, and Kui Ren. WMCopier: Forging invisible watermarks on arbitrary images. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2026.

[71] 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.

[72] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. SirenAttack: Generating adversarial audio for end-to-end acoustic systems. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2020.

[73] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*, 2006.

[74] ElevenLabs. ElevenLabs free AI voice generator. URL https://elevenlabs.io/.

[75] Sandy Engelhardt, Lalith Sharan, Matthias Karck, Raffaele De Simone, and Ivo Wolf. Cross-domain conditional generative adversarial networks for stereoscopic hyperrealism in surgical training. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2019.

[76] Gautier Evennou and Ewa Kijak. The forensic cost of watermark removal: From dedicated attacks to image editing. *arXiv preprint arXiv:2604.25491*, 2026.

[77] Shanglun Feng and Florian Tramèr. Privacy backdoors: Stealing data with corrupted pretrained models. In *International Conference on Machine Learning (ICML)*, 2024.

[78] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[79] First Direct. What is voice ID security? URL https://www1.firstdirect.com/banking/ways-to-bank/telephone-banking/#voice-id-security.

[80] Joel Frank and Lea Schönherr. WaveFake: A data set to facilitate audio deepfake detection. In *Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

[81] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *International Conference on Machine Learning (ICML)*, 2020.

[82] Norman R. French and John C. Steinberg. Factors governing the intelligibility of speech sounds. *The Journal of the Acoustical Society of America*, 1947.

[83] Wanying Ge, Michele Panariello, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Partially-connected differentiable architecture search for deepfake and spoofing detection. In *Interspeech*, 2021.

[84] Wanying Ge, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Raw differentiable architecture search for speech deepfake and spoofing detection. In *ASVspoof 2021 Workshop*, 2021.

[85] Alejandro Gomez-Alanis, Jose A. Gonzalez-Lopez, and Antonio M. Peinado. Adversarial transformation of spoofing attacks for voice biometrics. In *IberSPEECH*, 2021.

[86] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[87] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[88] Google. What is generative AI and what are its applications? URL https://cloud.google.com/use-cases/generative-ai.

[89] Google. Google Cloud Speech-to-Text. URL https://cloud.google.com/speech-to-text.

[90] Yevin Nikhel Goonatilake and Giuseppe Ateniese. Removing the watermark is not enough: Forensic stealth in generative-AI watermark removal. *arXiv preprint arXiv:2605.09203*, 2026.

[91] Sven Gowal and Pushmeet Kohli. Identifying AI-generated images with SynthID. *Google DeepMind Blog*, 2023. URL https://deepmind.google/discover/blog/identifying-ai-generated-images-with-synthid/.

[92] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[93] 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 *Workshop on Artificial Intelligence and Security (AISec), at CCS*, 2023.

[94] Tom Groenfeldt. Citi uses voice prints to authenticate customers quickly and effortlessly. *Forbes*, 2016. URL https://www.forbes.com/sites/tomgroenfeldt/2016/06/27/citi-uses-voice-prints-to-authenticate-customers-quickly-and-effortlessly/.

[95] Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. In *International Conference on Learning Representations (ICLR)*, 2025.

[96] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations (ICLR)*, 2018.

[97] Krishnaprasad Hande. General availability: Azure confidential vms with NVIDIA H100 tensor core GPUs. *Azure Confidential Computing Blog*, 2024. URL https://techcommunity.microsoft.com/blog/azureconfidentialcomputingblog/general-availability-azure-confidential-vms-with-nvidia-h100-tensor-core-gpus/4242644.

[98] Qingying Hao, Licheng Luo, Steve T. K. Jan, and Gang Wang. It's not what it looks like: Manipulating perceptual hashing based applications. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021.

[99] Drew Harwell. Scarlett Johansson on fake AI-generated sex videos: "nothing can stop someone from cutting and pasting my image". *The Washington Post*, 2018. URL https://www.washingtonpost.com/technology/2018/12/31/scarlett-johansson-fake-ai-generated-sex-videos-nothing-can-stop-someone-cutting-pasting-my-image/.

[100] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[101] Melissa Heikkilä. Google DeepMind has launched a watermarking tool for AI-generated images. *MIT Technology Review*, 2023. URL https://www.technologyreview.com/2023/08/29/1078620/google-deepmind-has-launched-a-watermarking-tool-for-ai-generated-images/.

[102] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[103] Mitch Hill, Jonathan Craig Mitchell, and Song-Chun Zhu. Stochastic security: Adversarial defense using long-run dynamics of energy-based models. In *International Conference on Learning Representations (ICLR)*, 2020.

[104] Emmie Hine and Luciano Floridi. New deepfake regulations in China are a tool for social stability, but at what cost? *Nature Machine Intelligence*, 2022.

[105] Chih-Hui Ho and Nuno Vasconcelos. DISCO: Adversarial defense with local implicit functions. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[106] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[107] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[108] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q. Weinberger. A new defense against adversarial images: Turning a weakness into a strength. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[109] Shu Hu, Yuezun Li, and Siwei Lyu. Exposing GAN-generated faces using inconsistent corneal specular highlights. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[110] Yuepeng Hu, Zhengyuan Jiang, Moyang Guo, and Neil Zhenqiang Gong. A transfer attack to image watermarks. In *International Conference on Learning Representations (ICLR)*, 2025.

[111] Guang Hua, Andrew Beng Jin Teoh, and Haijian Zhang. Towards end-to-end synthetic speech detection. *IEEE Signal Processing Letters*, 2021.

[112] Hua Hua, Ziyi Chen, Yuxiang Zhang, Ming Li, and Pengyuan Zhang. Improving spoofing capability for end-to-end any-to-many voice conversion. In *International Workshop on Deepfake Detection for Audio Multimedia (DDAM)*, 2022.

[113] Hydrogenaudio. Pre-emphasis. URL https://wiki.hydrogenaud.io/index.php?title=Pre-emphasis.

[114] ID R&D. ID R&D Voice Anti-Spoofing. URL https://www.idrnd.ai/voice-anti-spoofing.

[115] ID R&D. ID R&D ranks first in detecting synthetic speech in global ASVspoof challenge, 2019. URL https://www.idrnd.ai/id-rd-ranks-first-in-detecting-synthetic-speech-in-global-asvspoof-challenge/.

[116] Anubhav Jain, Yuya Kobayashi, Naoki Murata, Yuhta Takida, Takashi Shibuya, Yuki Mitsufuji, Niv Cohen, Nasir Memon, and Julian Togelius. Exposing vulnerabilities in latent-noise diffusion watermarks, 2025. URL https://openreview.net/forum?id=yGNLtD7VQ2.

[117] Arindam Jati, Chin-Cheng Hsu, Monisankha Pal, Raghuveer Peri, Wael AbdAlmageed, and Shrikanth Narayanan. Adversarial attack and defense strategies for deep speaker recognition systems. *Computer Speech & Language*, 2021.

[118] Yonghyun Jeong, Doyeon Kim, Seungjai Min, Seongho Joe, Youngjune Gwon, and Jongwon Choi. BiHPF: Bilateral high-pass filters for robust deepfake detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[119] Yi Jiang and Dengpan Ye. Black-box adversarial attacks against audio forensics models. *Security and Communication Networks*, 2022.

[120] Zhengyuan Jiang, Jinghuai Zhang, and Neil Zhenqiang Gong. Evading watermark-based detection of AI-generated content. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023.

[121] Sonal Joshi, Jesús Villalba, Piotr Żelasko, Laureano Moro-Velázquez, and Najim Dehak. Study of pre-processing defenses against adversarial attacks on state-of-the-art speaker recognition systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[122] 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*, 2021.

[123] Jee-weon Jung, Hee-Soo Heo, Hemlata Tak, Hye-jin Shim, Joon Son Chung, Bong-Jin Lee, Ha-Jin Yu, and Nicholas Evans. AASIST: Audio anti-spoofing using integrated spectro-temporal graph attention networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[124] Mintong Kang, Dawn Song, and Bo Li. DiffAttack: Evasion attacks against diffusion-based adversarial purification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[125] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

[126] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[127] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[128] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[129] Andre Kassis and Urs Hengartner. Breaking security-critical voice authentication. In *IEEE Symposium on Security and Privacy (SP)*, 2023.

[130] Andre Kassis and Urs Hengartner. UnMarker: A universal attack on defensive image watermarking. In *IEEE Symposium on Security and Privacy (SP)*, 2025.

[131] Andre Kassis, Urs Hengartner, and Yaoliang Yu. DiffBreak: Is diffusion-based purification robust? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[132] Louis Kerner, Michel Meintz, Bihe Zhao, Franziska Boenisch, and Adam Dziedzic. BitMark: Watermarking bitwise autoregressive image generative models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[133] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[134] Tomi Kinnunen, Jaime Lorenzo-Trueba, Junichi Yamagishi, Tomoki Toda, Daisuke Saito, Fernando Villavicencio, and Zhenhua Ling. A spoofing benchmark for the 2018 voice conversion challenge: Leveraging from spoofing countermeasures for speech artifact assessment. In *Odyssey: The Speaker and Language Recognition Workshop*, 2018.

[135] Jan Kociszewski, Hubert Jastrzębski, Tymoteusz Stępkowski, Filip Manijak, Krzysztof Rojek, Franziska Boenisch, and Adam Dziedzic. SERUM: Simple, efficient, robust, and

unifying marking for diffusion-based image generation. In *International Conference on Learning Representations (ICLR)*, 2026.

[136] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[137] Xiaoxi Kong, Jieyu Yuan, Pengdi Chen, Yuanlin Zhang, Chongyi Li, and Bin Li. MarkCleaner: High-fidelity watermark removal via imperceptible micro-geometric perturbation. *arXiv preprint arXiv:2602.01513*, 2026.

[138] Felix Kreuk, Yossi Adi, Moustapha Cisse, and Joseph Keshet. Fooling end-to-end speaker verification with adversarial examples. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[139] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[140] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2017.

[141] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*. Chapman and Hall/CRC, 2018.

[142] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[143] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations (ICLR)*, 2021.

[144] Vejay Lalla, Adine Mitrani, and Zach Harned. Artificial intelligence: Deepfakes in the entertainment industry. *WIPO Magazine*, 2022. URL https://www.wipo.int/wipo_magazine/en/2022/02/article_0003.html.

[145] Yee Wah Lau, Michael Wagner, and Dat Tran. Vulnerability of speaker verification to voice mimicking. In *International Symposium on Intelligent Multimedia, Video and Speech Processing (ISIMP)*, 2004.

[146] Galina Lavrentyeva, Sergey Novoselov, Andzhukaev Tseren, Marina Volkova, Artem Gorlanov, and Alexandr Kozlov. STC antispoofing systems for the ASVspoof2019 challenge. In *Interspeech*, 2019.

[147] 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 (SP)*, 2019.

[148] Minjong Lee and Dongwoo Kim. Robust evaluation of diffusion-based adversarial purification. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[149] Kecen Li, Zhicong Huang, Xinwen Hou, and Cheng Hong. GaussMarker: Robust dual-domain watermark for diffusion models. In *International Conference on Machine Learning (ICML)*, 2025.

[150] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, and B. Guo. Face X-Ray for more general face forgery detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[151] Xu Li, Na Li, Jinghua Zhong, Xixin Wu, Xunying Liu, Dan Su, Dong Yu, and Helen Meng. Investigating robustness of adversarial samples detection for automatic speaker verification. In *Interspeech*, 2020.

[152] Xu Li, Jinghua Zhong, Xixin Wu, Jianwei Yu, Xunying Liu, and Helen Meng. Adversarial attacks on GMM i-Vector based speaker verification systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[153] Xu Li, Na Li, Chao Weng, Xunying Liu, Dan Su, Dong Yu, and Helen Meng. Replay and synthetic speech detection with Res2Net architecture. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[154] Xu Li, Xixin Wu, Hui Lu, Xunying Liu, and Helen Meng. Channel-wise gated Res2Net: Towards robust detection of synthetic speech attacks. In *Interspeech*, 2021.

[155] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[156] Zhuohang Li, Cong Shi, Yi Xie, Jian Liu, Bo Yuan, and Yingying Chen. Practical adversarial attacks against speaker recognition systems. In *ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2020.

[157] Zhuohang Li, Yi Wu, Jian Liu, Yingying Chen, and Bo Yuan. AdvPulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[158] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[159] Shaohui Liu, Yi Wei, Jiwen Lu, and Jie Zhou. An improved evaluation framework for generative adversarial networks. *arXiv preprint arXiv:1803.07474*, 2018.

[160] Songxiang Liu, Haibin Wu, Hung-yi Lee, and Helen Meng. Adversarial attacks on spoofing countermeasures of automatic speaker verification. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[161] Xuechen Liu, Xin Wang, Md Sahidullah, Jose Patino, Héctor Delgado, Tomi Kinnunen, Massimiliano Todisco, Junichi Yamagishi, Nicholas Evans, Andreas Nautsch, and Kong Aik Lee. ASVspoof 2021: Towards spoofed and deepfake speech detection in the wild. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[162] Yiming Liu, Kezhao Liu, Yao Xiao, ZiYi Dong, Xiaogang Xu, Pengxu Wei, and Liang Lin. Towards understanding the robustness of diffusion-based purification: A stochastic perspective. In *International Conference on Learning Representations (ICLR)*, 2025.

[163] Keane Lucas, Matthew Jagielski, Florian Tramèr, Lujo Bauer, and Nicholas Carlini. Randomness in ML defenses helps persistent attackers and hinders evaluators. *arXiv preprint arXiv:2302.13464*, 2023.

[164] Nils Lukas and Florian Kerschbaum. PTW: Pivotal tuning watermarking for pre-trained image generators. In *USENIX Security Symposium (USENIX Security)*, 2023.

[165] 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.

[166] Hongwei Luo, Yijie Shen, Feng Lin, and Guoai Xu. Spoofing speaker verification system by adversarial examples leveraging the generalized speaker difference. *Security and Communication Networks*, 2021.

[167] 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.

[168] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do GANs leave artificial fingerprints? In *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2019.

[169] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *IEEE Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2019.

[170] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In *Python in Science Conference (SciPy)*, 2015.

[171] Dongyu Meng and Hao Chen. MagNet: A two-pronged defense against adversarial examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[172] Yan Meng, Jiachun Li, Matthew Pillari, Arjun Deopujari, Liam Brennan, Hafsah Shamsie, Haojin Zhu, and Yuan Tian. Your microphone array retains your identity: A robust voice liveness detection system for smart speaker. In *USENIX Security Symposium (USENIX Security)*, 2022.

[173] Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 2023.

[174] Microsoft. Azure Speech-to-Text. URL https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text/.

[175] Dan Milmo. UK engineering firm Arup falls victim to £20m deepfake scam. *The Guardian*, 2024. URL https://www.theguardian.com/technology/article/2024/may/17/uk-engineering-arup-deepfake-scam-hong-kong-ai-video?

[176] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning (ICML)*, 2018.

[177] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. All your voices are belong to us: Stealing voices to fool humans and machines. In *European Symposium on Research in Computer Security (ESORICS)*, 2015.

[178] Andreas Müller, Denis Lukovnikov, Jonas Thietke, Asja Fischer, and Erwin Quiring. Black-box forgery attacks on semantic watermarks for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[179] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. VoxCeleb: A large-scale speaker identification dataset. In *Interspeech*, 2017. Dataset: https://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox1.html.

[180] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. Adversary instantiation: Lower bounds for differentially private machine learning. In *IEEE Symposium on Security and Privacy (SP)*, 2021.

[181] 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. *arXiv preprint arXiv:2510.09023*, 2025.

[182] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Animashree Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning (ICML)*, 2022.

[183] Nuance. Gatekeeper: Cloud-Native Biometric Security for Every Channel. URL https://www.nuance.com/omni-channel-customer-engagement/authentication-and-fraud-prevention/gatekeeper.html.

[184] Nuance. Nuance Text-to-Speech. URL https://www.nuance.com/omni-channel-customer-engagement/voice-and-ivr/text-to-speech.html.

[185] Nuance. Nuance VocalPassword: Voice biometrics authentication. URL https://www.nuance.com/content/dam/nuance/en_us/collateral/enterprise/brochure/br-vocalpassword-en-us.pdf.

[186] Nuance. ING introduces a voice-controlled mobile banking app powered by Nuance, 2014. URL https://news.nuance.com/2014-09-16-ING-Introduces-a-Voice-Controlled-Mobile-Banking-App-Powered-by-Nuance.

[187] Nuance. Nuance voice biometrics quickly identifies customers and strengthens security for national australia bank, 2020. URL https://news.nuance.com/2020-05-27-Nuance-Voice-Biometrics-Quickly-Identifies-Customers-and-Strengthens-Security-for-National-Australia-Bank.

[188] NVIDIA. Confidential compute on NVIDIA hopper H100, 2023. URL https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/HCC-Whitepaper-v1.0.pdf.

[189] Cynthia Lopez Olson. Social engineering attacks by the numbers: Prevalence, costs, and impact. *Datafloq*, 2019. URL https://datafloq.com/social-engineering-attacks-numbers-cost/.

[190] OpenAI. DALL·E 2. URL https://openai.com/dall-e-2.

[191] Opus Research. 2022 Intelligent Authentication and Fraud Prevention Intelliview. *Nuance*, 2022. URL https://www.nuance.com/content/dam/nuance/en_us/collateral/enterprise/report/ar-opus-2022-intelligent-authentication-and-fraud-prevention-intelliview-en-us.pdf.

[192] Yidong Ouyang, Liyan Xie, and Guang Cheng. Improving adversarial robustness through the contrastive-guided diffusion process. In *International Conference on Machine Learning (ICML)*, 2023.

[193] Monisankha Pal, Arindam Jati, Raghuveer Peri, Chin-Cheng Hsu, Wael AbdAlmageed, and Shrikanth Narayanan. Adversarial defense for deep speaker recognition using hybrid adversarial training. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[194] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2019.

[195] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2016.

[196] 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)*, 2017.

[197] Hemant A. Patil and Madhu R. Kamble. A survey on replay attack detection for automatic speaker verification (ASV) system. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018.

[198] ShengYun Peng, Weilin Xu, Cory Cornelius, Matthew Hull, Kevin Li, Rahul Duggal, Mansi Phute, Jason Martin, and Duen Horng Chau. Robust principles: Architectural design principles for adversarially robust CNNs. In *British Machine Vision Conference (BMVC)*, 2023.

[199] Yanchun Peng, Na Wang, Chuan Qin, Guorui Feng, and Xinpeng Zhang. INNA: An invertible neural network-based attack framework for robust image watermarking. *Information Fusion*, 2026.

[200] S. Pereira and T. Pun. Robust template matching for affine resistant image watermarks. *IEEE Transactions on Image Processing*, 2000.

[201] Billy Perrigo. Facebook and YouTube struggle to remove new zealand shooting. *TIME*, 2019. URL https://time.com/5552367/new-zealand-shooting-video-facebook-youtube-twitter/.

[202] Billy Perrigo. Why tech companies struggled to stop buffalo terror video. *TIME*, 2022. URL https://time.com/6177640/buffalo-shooting-twitch-social-media/.

[203] Pindrop Security. Deepfake Detection Technology. URL https://www.pindrop.com/solution/deepfake-detection/.

[204] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[205] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *European Conference on Computer Vision (ECCV)*, 2020.

[206] Javier Rando, Jie Zhang, Nicholas Carlini, and Florian Tramèr. Adversarial ML problems are getting harder to solve and to evaluate. *arXiv preprint arXiv:2502.02260*, 2025.

[207] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Data augmentation can improve robustness. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[208] Resemble AI. Resemblyzer. URL https://github.com/resemble-ai/Resemblyzer.

[209] Rev. Microsoft Azure speech recognition vs. Rev AI speech to text API. URL https://www.rev.com/blog/resources/microsoft-azure-speech-recognition-vs-rev-ai-speech-to-text-api.

[210] Mauro Ribeiro, Katarina Grolinger, and Miriam A.M. Capretz. MLaaS: Machine learning as a service. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2015.

[211] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[212] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[213] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In *International Conference on Machine Learning (ICML)*, 2019.

[214] Mehrdad Saberi, Vinu Sankar Sadasivan, Keivan Rezaei, Aounon Kumar, Atoosa Chegini, Wenxiao Wang, and Soheil Feizi. Robustness of AI-image detectors: Fundamental limits and practical attacks. In *International Conference on Learning Representations (ICLR)*, 2024.

[215] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[216] Tim Sainburg. timsainb/noisereduce: v1.0. URL https://doi.org/10.5281/zenodo.3243139.

[217] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J. Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[218] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

[219] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[220] Fahad Shamshad, Nils Lukas, and Karthik Nandakumar. RAVEN: Erasing invisible watermarks via novel view synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[221] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Gotta catch 'em all: Using honeypots to catch adversarial attacks on neural networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[222] Shawn Shan, Wenxin Ding, Emily Wenger, Haitao Zheng, and Ben Y. Zhao. Post-breach recovery: Protection against white-box adversarial examples for leaked DNN models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[223] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerrv-Ryan, et al. Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[224] ShieldMnt. Invisible watermark. URL https://github.com/ShieldMnt/invisible-watermark.

[225] Maliheh Shirvanian, Manar Mohammed, Nitesh Saxena, and S. Abhishek Anand. Voicefox: Leveraging inbuilt transcription to enhance the security of machine-human speaker verification against voice synthesis attacks. In *Annual Computer Security Applications Conference (ACSAC)*, 2020.

[226] David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A music, speech, and noise corpus. *arXiv preprint arXiv:1510.08484*, 2015.

[227] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[228] Kaiyu Song, Hanjiang Lai, Yan Pan, and Jian Yin. MimicDiffusion: Purifying adversarial perturbation via mimicking clean diffusion model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[229] 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.

[230] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[231] Vera Sorin, Yiftach Barash, Eli Konen, and Eyal Klang. Creating artificial images for radiology applications using generative adversarial networks (GANs)—a systematic review. *Academic Radiology*, 2020.

[232] James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 2002.

[233] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[234] Hemlata Tak, Jee-weon Jung, Jose Patino, Madhu Kamble, Massimiliano Todisco, and Nicholas Evans. End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection. In *ASVspoof 2021 Workshop*, 2021.

[235] Hemlata Tak, Massimiliano Todisco, Xin Wang, Jee-weon Jung, Junichi Yamagishi, and Nicholas Evans. Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation. In *Odyssey: The Speaker and Language Recognition Workshop*, 2022.

[236] Matthew Tancik, Ben Mildenhall, and Ren Ng. StegaStamp: Invisible hyperlinks in physical photographs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[237] TD Bank. Voice Print System. URL https://www.td.com/ca/products-services/investing/td-direct-investing/trading-platforms/voice-print-system-enroll.jsp.

[238] The Insight Partners. Voice biometrics market size worth $4.82bn, globally, by 2028. *PR Newswire*, 2022. URL https://www.prnewswire.com/news-releases/voice-biometrics-market-size-worth-4-82bn-globally-by-2028-at-20-6-cagr---exclusive-report-by-the-insight-partners-301552781.html.

[239] Tomoki Toda, Alan W. Black, and Keiichi Tokuda. Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Transactions on Audio, Speech, and Language Processing*, 2007.

[240] Massimiliano Todisco, Héctor Delgado, and Nicholas W. D. Evans. A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients. In *Odyssey: The Speaker and Language Recognition Workshop*, 2016.

[241] Massimiliano Todisco, Xin Wang, Ville Vestman, Md Sahidullah, Héctor Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi Kinnunen, and Kong Aik Lee. ASVspoof 2019: Future horizons in spoofed and fake audio detection. In *Interspeech*, 2019.

[242] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[243] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[244] Twilio. Conversational AI and APIs for SMS, Email, Voice. URL https://www.twilio.com/.

[245] US Bank. How do i enable or disable my biometrics when i log in? URL https://www.usbank.com/customer-service/knowledge-base/KB0069639.html.

[246] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[247] Jesús Villalba, Yuekai Zhang, and Najim Dehak. x-vectors meet adversarial attacks: Benchmarking adversarial robustness in speaker verification. In *Interspeech*, 2020.

[248] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[249] Chunpeng Wang, Qixian Hao, Shujiang Xu, Bin Ma, Zhiqiu Xia, Qi Li, Jian Li, and Yun-Qing Shi. RD-IWAN: Residual dense based imperceptible watermark attack network. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2022.

[250] Jinyi Wang, Zhaoyang Lyu, Dahua Lin, Bo Dai, and Hongfei Fu. Guided diffusion model for adversarial purification. *arXiv preprint arXiv:2205.14969*, 2022.

[251] Kaibo Wang, Xiaowen Fu, Yuxuan Han, and Yang Xiang. DiffHammer: Rethinking the robustness of diffusion-based adversarial purification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[252] Qing Wang, Pengcheng Guo, Sining Sun, Lei Xie, and John H. L. Hansen. Adversarial regularization for end-to-end robust speaker verification. In *Interspeech*, 2019.

[253] Qing Wang, Pengcheng Guo, and Lei Xie. Inaudible adversarial perturbations for targeted attack in speaker recognition. In *Interspeech*, 2020.

[254] Shu Wang, Jiahao Cao, Xu He, Kun Sun, and Qi Li. When the differences in frequency domain are compensated: Understanding and defeating modulated replay attacks on automatic speech recognition. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[255] Xin Wang and Junichi Yamagishi. A comparative study on recent neural spoofing countermeasures for synthetic speech detection. In *Interspeech*, 2021.

[256] Xin Wang, Junichi Yamagishi, Massimiliano Todisco, Héctor Delgado, Andreas Nautsch, Nicholas Evans, Md Sahidullah, Ville Vestman, Tomi Kinnunen, Kong Aik Lee, et al. ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Computer Speech & Language*, 2020.

[257] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. In *International Conference on Machine Learning (ICML)*, 2023.

[258] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

[259] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[260] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-Ring watermarks: Invisible fingerprints for diffusion images. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[261] Emily Wenger, Max Bronckers, Christian Cianfarani, Jenna Cryan, Angela Sha, Haitao Zheng, and Ben Y. Zhao. "Hello, It's Me": Deep learning-based speech synthesis attacks in the real world. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021.

[262] Kyle Wiggers. Google releases dataset to help AI systems spot fake audio recordings. *VentureBeat*, 2019. URL https://venturebeat.com/ai/google-releases-dataset-to-help-ai-systems-determine-if-an-audio-recording-is-real/.

[263] John Wiseman. py-webrtcvad: Python interface to the WebRTC voice activity detector (VAD). URL https://github.com/wiseman/py-webrtcvad.

[264] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[265] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations (ICLR)*, 2020.

[266] Haibin Wu, Songxiang Liu, Helen Meng, and Hung-yi Lee. Defense against adversarial attacks on spoofing countermeasures of ASV. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[267] Quanlin Wu, Hang Ye, and Yuntian Gu. Guided diffusion model for adversarial purification from random noise. *arXiv preprint arXiv:2206.10875*, 2022.

[268] Zhizheng Wu and Haizhou Li. On the study of replay and voice conversion attacks to text-dependent speaker verification. *Multimedia Tools and Applications*, 2016.

[269] Zhizheng Wu, Junichi Yamagishi, Tomi Kinnunen, Cemal Hanilçi, Mohammed Sahidullah, Aleksandr Sizov, Nicholas Evans, Massimiliano Todisco, and Héctor Delgado. ASVspoof: The automatic speaker verification spoofing and countermeasures challenge. *IEEE Journal of Selected Topics in Signal Processing*, 2017.

[270] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *International Conference on Learning Representations (ICLR)*, 2020.

[271] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[272] Chaowei Xiao, Zhongzhu Chen, Kun Jin, Jiongxiao Wang, Weili Nie, Mingyan Liu, Anima Anandkumar, Bo Li, and Dawn Song. DensePure: Understanding diffusion models towards adversarial robustness. In *Workshop on Trustworthy and Socially Responsible Machine Learning, at NeurIPS*, 2022.

[273] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations (ICLR)*, 2018.

[274] Yi Xie, Cong Shi, Zhuohang Li, Jian Liu, Yingying Chen, and Bo Yuan. Real-time, universal, and robust adversarial attacks against speaker recognition systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[275] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Network and Distributed System Security Symposium (NDSS)*, 2018.

[276] Xiaoyun Xu, Zhuoran Liu, Stefanos Koffas, Shujian Yu, and Stjepan Picek. BAN: Detecting backdoors activated by adversarial neuron noise. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[277] Junichi Yamagishi, Xin Wang, Massimiliano Todisco, Md Sahidullah, Jose Patino, Andreas Nautsch, Xuechen Liu, Kong Aik Lee, Tomi Kinnunen, Nicholas Evans, et al. ASVspoof 2021: Accelerating progress in spoofed and deepfake speech detection. In *ASVspoof 2021 Workshop*, 2021.

[278] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. ME-Net: Towards effective adversarial robustness with matrix estimation. In *International Conference on Machine Learning (ICML)*, 2019.

[279] Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[280] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial purification with score-based generative models. In *International Conference on Machine Learning (ICML)*, 2021.

[281] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[282] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry S. Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *International Conference on Learning Representations (ICLR)*, 2022.

[283] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A. Gunter. CommanderSong: A systematic approach for practical adversarial voice recognition. In *USENIX Security Symposium (USENIX Security)*, 2018.

[284] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016.

[285] Heiga Zen, Rob Clark, Ron J. Weiss, Viet Dang, Ye Jia, Yonghui Wu, Yu Zhang, and Zhifeng Chen. LibriTTS: A corpus derived from LibriSpeech for text-to-speech. In *Interspeech*, 2019.

[286] Boya Zhang, Weijian Luo, and Zhihua Zhang. Purify++: Improving diffusion-purification with advanced diffusion models and control of randomness. *arXiv preprint arXiv:2310.18762*, 2023.

[287] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.

[288] Jiawei Zhang, Zhongzhu Chen, Huan Zhang, Chaowei Xiao, and Bo Li. DiffSmooth: Certifiably robust learning via diffusion models and local smoothing. In *USENIX Security Symposium (USENIX Security)*, 2023.

[289] Linghan Zhang, Sheng Tan, Jie Yang, and Yingying Chen. VoiceLive: A phoneme localization based liveness detection for voice authentication on smartphones. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[290] Linghan Zhang, Sheng Tan, and Jie Yang. Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[291] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[292] Xingyu Zhang, Xiongwei Zhang, Wei Liu, Xia Zou, Meng Sun, and Jian Zhao. Waveform level adversarial example generation for joint attacks against both automatic speaker verification and spoofing countermeasures. *Engineering Applications of Artificial Intelligence*, 2022.

[293] Xingyu Zhang, Xiongwei Zhang, Xia Zou, Haibo Liu, and Meng Sun. Towards generating adversarial examples on combined systems of automatic speaker verification and spoofing countermeasure. *Security and Communication Networks*, 2022.

[294] You Zhang, Fei Jiang, and Zhiyao Duan. One-class learning towards synthetic voice spoofing detection. *IEEE Signal Processing Letters*, 2021.

[295] Yuekai Zhang, Ziyan Jiang, Jesús Villalba, and Najim Dehak. Black-box attacks on spoofing countermeasures using transferability of adversarial examples. In *Interspeech*, 2020.

[296] Yuxiang Zhang, Wenchao Wang, and Pengyuan Zhang. The effect of silence and dual-band fusion in anti-spoofing system. In *Interspeech*, 2021.

[297] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[298] Xuandong Zhao, Kexun Zhang, Zihao Su, Saastha Vasan, Ilya Grishchenko, Christopher Kruegel, Giovanni Vigna, Yu-Xiang Wang, and Lei Li. Invisible image watermarks

are provably removable using generative AI. In *Workshop on Challenges in Deploying Generative AI, at ICML*, 2023.

[299] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*, 2023.

[300] Baolin Zheng, Peipei Jiang, Qian Wang, Qi Li, Chao Shen, Cong Wang, Yunjie Ge, Qingyang Teng, and Shenyi Zhang. Black-box adversarial attacks on commercial speech platforms with minimal information. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021.

[301] Thomas Fang Zheng and Lantian Li. Robustness-related issues in speaker recognition. *SpringerBriefs in Signal Processing*, 2017.

[302] Hong Zhu, Shengzhi Zhang, and Kai Chen. AI-Guardian: Defeating adversarial attacks using backdoors. In *IEEE Symposium on Security and Privacy (SP)*, 2023.

[303] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding data with deep networks. In *European Conference on Computer Vision (ECCV)*, 2018.

[304] Zoom Support. Managing facial recognition check-in for visitors. URL https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0082683.

APPENDICES

Appendix A

Breaking Security-Critical Voice Authentication

A.1 ASV Models

Details of our chosen ASVs are in Table A.1. To use Amazon Connect Voice ID and Resemblyzer, we had to calibrate them to obtain decision thresholds for our specific use case. To calibrate Voice ID, first, we enrolled all the speakers in the evaluation subset of the ASVspoof2019 dataset using the process outlined in §3.5. Afterward, we selected 480 **bonafide** samples from the evaluation subset of ASVspoof2019 s.t. none of these were used for speaker enrollment. We denote this set as AWSCalibrate. For 240 of the samples, we invoked Voice ID with the ID of the actual speaker of the sample. For the other 240 samples, Voice ID was invoked with a different speaker's ID. The scores were used to calculate the EER threshold.

As for Resemblyzer, for each **bonafide** sample in the evaluation subset of ASVspoof2019, we performed verification with two bonafide reference samples; one that belongs to the same speaker, and another which belongs to a different speaker. The scores were used to calculate the threshold.

A.2 CM Calibration for The Over-Telephony Setting

Due to the costly nature of our over-telephony experiments, preventing us from transmitting thousands of samples over the phone, and the need to have well-substantiated results that

System	*ShadowASV*	ASVTarget1	*ASVTarget2*	*ASVTarget3*	*Google / Resemblyzer*	*AWS Voice ID*
Architechture	GMM-IVECTOR (Gaussians: 1024, IVECTOR Dim.: 300)	GMM-IVECTOR (Gaussians: 2048, IVECTOR Dim.: 400)	GMM-IVECTOR (Gaussians: 2048, IVECTOR Dim.: 400)	XVECTOR	DNN	N/A
Frontend	MFCC	MFCC	LPMS	MFCC	Mel Spectrogram	N/A
Training Set	ASVspoof2019-dev (bonafide only)	ASVspoof2019-train (bonafide only)	ASVspoof2019-train (bonafide only)	Voxceleb1-Train	Voxceleb1 & Voxceleb2 & LibriSpeech	N/A
Calibration / Eval Set	ASVspoof2019-eval (bonafide only)	ASVspoof2019-eval (bonafide only)	ASVspoof2019-eval (bonafide only)	Voxceleb1-Test	ASVspoof2019-eval (bonafide only)	AWSCalibrateCM
EER	2.76%	3%	4.65%	6.42%	13%	6.45%

Table A.1: Architectures, datasets, and EERs of the different ASVs with respect to their evaluation/calibration datasets.

Frontend		LPMS	LFCC				CQT			Raw Waveform					
System / *Dataset*		LCNN	AIR	AIR_AM	DARTS	LCNN_LSTM	MCG	MLCG	RES2NET	SSNET	RAWGAT_ST	RAWDARTS	AASIST	AASIST-L	RAWNET2 (BASELINE)
ASVspoof2021	*EER*	29.37%	18.23%	19.93%	17.16%	17.07%	**11.81%**	18.16%	**12.7%**	**13.8%**	**8.09%**	**11.55%**	**8.67%**	**10.53%**	**9.5%**
	Thresh.	1.56	-0.87	0.21	2.94	0.00	0.57	0.12	0.09	1.15	-2.47	0.01	-2.7	-0.45	N/A
AWSCalibrateCM	*EER*	N/A	N/A	N/A	N/A	N/A	10.95%	N/A	**10.52%**	**10.41%**	**4.77%**	**6.61%**	**5.74%**	**4.33%**	N/A
	Thresh.						0.98		0.31	2.06	-1.35	0.00	-1.44	-0.45	

Table A.2: CMs' EERs and their respective thresholds in the phone setting w.r.t. ASVspoof2021/AWSCalibrateCM. EERs below 15% are in bold.

warrant including the different CMs in this experiment, calibration was done in two phases. First, we use the ASVspoof2021 dataset [277], which evaluates the ability to deal with the channel effect. The results are in Table A.2 (first row). For reference, we include the EER (9.5%) for the baseline model in the ASVspoof2021 challenge ($RAWNET2$). The state-of-the-art WAV2VEC [235] is missing from the table, but as explained in §3.5, it is superior to all known models with a 0.8% EER. Performance is degraded significantly for some models due to the channel effect, while for others, the degradation is not as drastic. We filter-out models lacking generalization to the phone setting, leaving only those with EERs below 15%. While this is not practical, it is on par with the $RAWNET2$ baseline. The attacker also needs to bypass other components as well, further decreasing the actual success rates.

After choosing the models that are somewhat generalizable to the phone scenario, we empirically verify their ability to perform classification under our specific settings (i.e., when transmitting the samples using Twilio to Amazon Connect) as the different codecs used and system specifications may reveal that not all models are in fact applicable for this use case. We choose a subset, named AWSCalibrateCM, of spoofed and bonafide samples (the same 474 spoofed samples from §3.5 and 230 other bonafide samples) from the evaluation section of ASVspoof2019 and transmit them using Twilio to our Amazon Connect instance. We then extract the transmitted samples from the conversations we record at the receiver and use these samples to test whether the CMs can still differentiate between the bonafide and spoofed samples and calculate their alternative EERs (w.r.t. AWSCalibrateCM) accordingly. The results can be found in the second row of Table A.2 and prove the model's ability to generalize, achieving relatively low EERs. Furthermore, the thresholds for achieving these EERs (in parentheses) are higher, making the attacker's task ever harder (a higher score is needed to accept the sample). Hence, we use these EERs for our over-telephony experiment.

A.3 Shattering the Myth of Transferability

Our guiding principle is this: models for audio-related tasks must be robust to various types of naturally-occurring noise, the most common of which being additive (white). This is known to the anti-spoofing community [277]. Models lacking this robustness are untrustworthy and should not be deployed in real-world environments. Specifically, spoofed samples tend to be clean as the outputs of computer algorithms, whereas bonafide samples are recorded using microphones by humans and are more likely to exhibit natural noise (see §3.4.2). Transferability is useless unless it affects noise-robust systems.

There are currently a variety of models that have demonstrated robustness to noisy samples, making them suitable for real-world environments, such as Void [6] and WAV2VEC [235] (which was trained on noise-augmented data and with which we experiment in §3.6.1). This is trivially expected, especially from commercial systems used in practice. The participating systems in ASVspoof2019, being on par with industry standards (see §3.5), may very well be able to generalize to noisy settings. Yet, there is no guarantee that **all** these systems do, as they have not been evaluated in these settings. Previous work demonstrating adversarial attacks on CMs [160, 295, 293, 292] overlook this fact, which we assume to be the main reason behind their high transferability rates. Our hypothesis is that models that exhibit a significantly degraded performance under transferability are not robust to noise, enabling **any** noise to bring them out of balance and cause them to misclassify. Additionally, such

Frontend	LPMS	LFCC				CQT			Raw Waveform				
System / *Perturbation*	LCNN	AIR	AIR_AM	DARTS	LCNN_LSTM	MCG	MLCG	RES2NET	SSNET	RAWGAT_ST	RAWDARTS	AASIST	AASIST-L
None	3.16%	2.32%	3.59%	4.01%	2.95%	1.90%	4.01%	2.95%	2.74%	**3.16%**	**1.90%**	**2.32%**	2.32%
0.0005	9.28%	7.17%	8.44%	6.12%	3.80%	1.69%	5.49%	2.74%	6.54%	**3.16%**	**1.90%**	**2.32%**	2.32%
0.0015	9.28%	17.09%	14.98%	17.51%	4.85%	4.64%	**9.49%**	**6.54%**	14.98%	**3.16%**	**1.90%**	**2.32%**	**2.53%**
0.0025	9.28%	18.99%	**15.61%**	19.83%	5.06%	12.24%	6.96%	4.43%	**18.14%**	2.95%	**1.90%**	**2.32%**	**2.53%**
0.0035	**9.49%**	**20.25%**	15.19%	**20.46%**	**5.27%**	**18.35%**	6.33%	1.69%	17.09%	2.95%	**1.90%**	**2.32%**	**2.53%**
Best	**9.49%**	**20.25%**	**15.61%**	**20.46%**	**5.27%**	**18.35%**	**9.49%**	**6.54%**	**18.14%**	**3.16%**	**1.90%**	**2.32%**	**2.53%**

Table A.3: CM performance under additive noise. Bold numbers indicate the highest success rate against each system.

models are all overfitted to the same training data and therefore it would be natural that they would share many common decision boundaries, as opposed to robust models. Our goal is to determine which systems are not robust to additive noise and test its connection to transferability.

Our large-scale experiment is this: first, we introduce noise to all the samples in our evaluation dataset with various budgets. Then, we generate adversarial examples using common methods (*I-FGSM* [140], *PGD* [140], Carlini & Wagner ($C\ \&\ W$) [32]) and $LCNN$ (trained on the development subset of ASVspoof2019, as opposed to the $LCNN$ target in all of our tables, which was trained on the training subset— to enforce blackbox settings) as the shadow. This system was used in previous works, leading to high transferability [160].

The results are in Tables A.3 & A.4, respectively, proving a direct link between lacking noise robustness and transferability. $LCNN$ shares its architecture with the shadow, so the high success rates are expected (not fully blackbox). $LCNN_LSTM$ and $RES2NET$ exhibit good noise robustness and, while some transferability occurs for these, it remains limited. Systems that are highly noise-robust ($RAWGAT_ST$, $RAWDARTS$, $AASIST$, $AASIST$-L) are insensitive to transferability. The systems vulnerable to transferability (AIR, AIR_AM, $DARTS$, MCG, $MLCG$) are those lacking noise robustness and not suitable for practical use, proving transferability is not effective in practice.

A.4 Results for Individual Transformations

Individual results are in Table A.5. We reiterate the statement made in §3.6.1 noting that each transformation targets a specific set of heuristics and the dependence of the CMs on these principles varies according to their design and underlying features. Hence, the transformations individually do not always succeed in attacking all CMs. The universal

	Frontend	LPMS	LFCC				CQT			Raw Waveform				
Alg.	*System* / *Perturbation*	LCNN	AIR	AIR_AM	DARTS	LCNN_LSTM	MCG	MLCG	RES2NET	SSNET	RAWGAT_ST	RAWDARTS	AASIST	AASIST-L
None	None	3.16	2.32	3.59	4.01	2.95	1.90	4.01	2.95	2.74	3.16	1.90	2.32	2.32
I-FGSM	iters=5, eps=0.0005	23.00	18.14	15.61	18.78	7.17	9.70	12.03	**8.86**	8.86	**3.16**	**1.90**	**2.32**	2.32
	iters=5, eps=0.0015	30.80	30.38	21.10	22.15	6.96	27.43	16.24	2.95	17.51	2.95	**1.90**	**2.32**	2.53
	iters=5, eps=0.0025	41.56	39.03	23.84	22.36	5.49	34.39	23.63	0.42	17.30	2.95	**1.90**	**2.32**	2.53
	iters=5, eps=0.0035	47.26	44.51	26.79	23.36	4.22	41.14	**26.16**	0.21	16.67	2.74	**1.90**	**2.32**	2.74
	iters=5, eps=0.0045	49.37	50.63	28.69	23.00	3.80	37.13	24.47	0.21	16.67	2.53	**1.90**	1.90	2.95
	iters=5, eps=0.0055	51.48	**54.85**	**31.86**	**23.42**	3.59	**43.67**	25.74	0.00	16.24	2.74	**1.90**	1.90	**3.88**
	iters=10, eps=0.0015	47.68	35.23	23.00	22.36	**7.38**	28.27	15.40	2.32	**17.93**	2.95	**1.90**	**2.32**	2.53
	iters=15, eps=0.0015	**55.91**	36.50	23.63	22.15	7.17	29.32	14.56	2.32	**17.93**	2.95	**1.90**	**2.32**	2.53
C & W (lr=0.01)	iters=20, confidence=0.0	**22.15**	**31.01**	**17.93**	19.20	**2.32**	**25.95**	**27.85**	**2.53**	**11.60**	1.27	**1.05**	**0.84**	**1.05**
	iters=20, confidence=5.0	16.24	20.89	17.30	**19.41**	1.05	10.55	17.51	1.69	5.27	**1.69**	0.84	**0.84**	**1.05**
PGD (restarts=5)	iters=20, eps=0.003	**70.46**	**42.62**	**23.00**	**21.73**	**8.23**	**45.36**	**16.67**	1.05	**15.82**	2.32	**1.90**	**2.32**	**3.59**
	iters=20, eps=0.0003	25.53	12.87	14.56	18.14	6.33	7.17	9.07	**5.06**	9.28	**3.16**	**1.90**	**2.32**	2.32
Best	N/A	**70.46**	**54.58**	**31.86**	**23.42**	**8.23**	**43.67**	**27.85**	**8.86**	**17.93**	**3.16**	**1.90**	**2.32**	**3.59**

Table A.4: CM performance (%) under transferability. For each attack type, bold numbers correspond to the configuration with the highest success rate against each system.

nature of our full attack lies in its ability to suppress multiple identifiable cues simultaneously. $F_1 - F_7$, when employed together as shown in §3.6.1, suffice to achieve this goal. Individual evaluations, while helpful in bench-marking each of the techniques independently, cannot alone determine their suitability as part of the attack. We select transforms that show potential against several CMs even if they are not effective against all systems or lead to a slight degradation against some, as the existence of other methods in the full attack can make up for such minor losses. The transformations' order is somewhat deterministic (see §3.4.3). Hence, evaluating each transformation's actual potential as part of the attack is better done in a cumulative manner, as is in §3.6.1.

We can see that F_1 (silence replacement) is highly effective against almost all CMs. F_2 is not as universal as for three systems (AIR, $MLCG$, $RES2NET$) we observe a mild attack degradation w.r.t. the spoofing baseline. Yet, for the majority of systems, F_2 increases the attack's success (often considerably— MCG, $SSNET$, $RAWDARTS$, $AASIST\text{-}L$), warranting its inclusion. Similar to F_1, F_3 (spectrum center boosting), F_6 (noise reduction), and F_7 (ADVSR) drastically increase the success rates for several systems and should naturally be included. As for F_4 (local echo), the findings are similar to F_2, being effective especially against CQT-based systems. Finally, F_5 (pre-emphasis) is powerful against

Frontend	LPMS	LFCC				CQT			Raw Waveform				
Transformation \ *System*	LCNN	AIR	AIR_AM	DARTS	LCNN_LSTM	MCG	MLCG	RES2NET	SSNET	RAWGAT_ST	RAWDARTS	AASIST	AASIST-L
None	3.16%	2.32%	3.59%	4.01%	2.95%	1.90%	4.01%	2.95%	2.74%	3.16%	1.90%	2.32%	2.32%
F1	**13.71%**	15.61%	13.92%	13.08%	10.97%	12.66%	9.28%	4.85%	12.66%	**7.59%**	1.69%	**4.85%**	**5.27%**
F2	3.16%	2.11%	3.80%	4.22%	3.80%	5.06%	3.38%	1.05%	6.96%	4.85%	2.74%	2.95%	3.16%
F3	4.01%	11.81%	11.81%	6.12%	**21.10%**	6.75%	6.33%	6.33%	2.11%	2.95%	**2.95%**	2.95%	2.32%
F4	2.95%	2.95%	4.01%	4.22%	4.43%	10.97%	6.12%	3.38%	2.53%	3.38%	2.11%	2.53%	2.74%
F5	4.01%	4.64%	4.85%	4.43%	1.90%	3.16%	3.80%	2.11%	5.49%	3.38%	2.11%	3.38%	4.43%
F6	1.48%	1.90%	4.64%	4.22%	3.59%	**15.40%**	6.12%	3.59%	3.59%	2.32%	1.90%	2.95%	4.22%
F7	9.28%	**29.32%**	**27.00%**	**18.57%**	11.18%	8.65%	**16.67%**	**8.65%**	**17.51%**	2.95%	1.90%	2.32%	2.74%

Table A.5: Independent transformations' success rates against CMs. Bold entries represent the best results for each system.

Frontend	LPMS	LFCC				CQT			Raw Waveform				
System	LCNN	AIR	AIR_AM	DARTS	LCNN_LSTM	MCG	MLCG	RES2NET	SSNET	RAWGAT_ST	RAWDARTS	AASIST	AASIST-L
ASR	0.63%	13.5%	9.07%	0.00%	2.53%	35.65%	9.7%	13.92%	13.5%	10.97%	10.55%	9.49%	12.03%

Table A.6: Results of a simple pre-emphasis attack with a large coefficient (0.97).

raw waveform-based systems ($SSNET, AASIST\text{-}L$, specifically). That said, its potential appears limited compared to when integrated into the full attack (see Table 3.2).

To fathom this, recall that we use a small pre-emphasis coefficient (0.5). As noted in §3.6.1 and demonstrated in Table A.6, F_5 is considerably stronger upon increasing the coefficient to 0.97. In the collective attack, F_5 is coupled with F_3, which itself similarly performs amplification in higher frequencies, serving as a precursor that compensates for using a smaller coefficient, making F_5's effects dominant again. We observe similar results for other transforms that are more powerful when applied jointly, such as F_2 when preceded by F_1 (see Table 3.2). The reasoning is similar, as there is a direct connection between these transformations and while one individually eliminates some machine cues, it requires others' assistance to achieve its full potential.

A.5 Content Preservation

We use the Python API of Google Cloud STT [89] and the REST API of Microsoft Azure STT [174]. For Google, we use the "video" model for transcription, as it is recommended for inputs with a 16KHz sampling rate, as in our case.

Our attacks assume state-of-the-art spoofing algorithms, such as Shen et al.'s [223] are accessible to the attacker. Such algorithms are known to produce high-quality speech and we assume they can generate speech that is transcribed correctly by STTs. Thus, we randomly select 25 spoofed samples from our high-quality evaluation dataset of 474. For each sample we generate an adversarial using our full attack. We test whether the STTs transcribe both samples identically. The common metric for evaluating STTs is the word error rate (WER), which is the percentage of words transcribed incorrectly in a given input on average. Since STTs are not perfect and make mistakes often, we cannot assume that if the provided input does not exactly transcribe to the requested text then it will be rejected. Instead, the system must account for its own mistakes. Thus, instead of strictly requiring that the two samples (spoofed and adversarial) transcribe exactly to the same text, we consider the adversarial attack successful if the ratio of different words is below the average WER for each of the STTs. We get the WER values from a benchmark [209].

Results. We achieve an attack success rate of 96% against Microsoft Azure STT and 76% against Google Cloud STT.

Appendix B

UnMarker: A Universal Attack on Defensive Image Watermarking

B.1 *UnMarker*'s Hyperparameters

High-frequency Disruptions (Stage 1— §4.5.1). We use *Alex* [291] for small images (128×128) and *DVL* [57] for larger sizes as the perceptual loss (ℓ_p). As we maximize *DFL*, we set t_{ℓ_p} to the largest value at which similarity and quality are retained. This depends on ℓ_p and the image size. For small images optimized using *Alex*, this optimal point is t_{Alex}=$5{\cdot}10^{-4}$. Since larger images require a different loss (*DVL*), similar thresholds do not indicate the same degree of resemblance. For 256×256 images, we find t_{DVL}=$4{\cdot}10^{-2}$ best achieves the above goals, while for larger images we have t_{DVL}=10^{-4}. Next, we select the initial c_{ℓ_p} that balances the spectral and visual losses. We want this starting value to be significant to assign a large weight to ℓ_p and ensure visual similarity is not violated, after which it can be decreased to allow the spectral loss to be more dominant while still ensuring this property. This initial setting only affects the time until it converges and must be empirically determined to speed up the process. We found an initial c_{Alex}=10^6 optimal for small images. For *DVL*, this was set at c_{DVL}=10^6 when the size is above 256×256 as these images require the *DVL* loss to be extremely small (t_{DVL}=10^{-4}), while for 256×256 images, it can be far smaller (c_{DVL}=10^{-2}).

The $\|\|_2$ component prevents unwanted peaks (see §4.5.1). Its threshold depends on the size and chosen *LPIPS*. While the *LPIPS* thresholds guarantee visual similarity to

the extent that the images will look similar up to, potentially, such peaks, $\|\|_2$'s thresholds must eliminate these singular points when the *LPIPS* thresholds are met. We empirically selected these $\|\|_2$ thresholds: for 128×128, we have $t_{\|\|_2} = 10^{-4}$, while 256×256 images allow $t_{\|\|_2} = 3 \cdot 10^{-5}$, and larger images operate at $t_{\|\|_2} = 10^{-4}$. We report the normalized thresholds (divided by the image size). In practice, they increase with the size since larger images accommodate stronger perturbations without visible cues. $c_{\|\|_2}$ multiplies this $\|\|_2$ when its threshold is violated to strongly steer the procedure away from these boundaries. This is not determined through a binary search, as we want it to have the same effect constantly, without undoing the previous iterations' effects. By tracking the losses, we arrive at $c_{\|\|_2} = 0.6$.

By monitoring the spectral loss, we can determine when it stops increasing for each value of c_{ℓ_p} and the number of steps after which updating c_{ℓ_p} does not result in improvements. This yields 5000 iterations for images smaller or larger than 256×256, and 2000 otherwise. The number of binary steps is 2 for sizes $\leqslant$ 256×256 and 4 otherwise. The optimizer is Adam with learning rate $2 \cdot 10^{-4}$, reduced by 0.5 when no improvement (above 10^{-4}) occurs for 10 iterations. We clip gradients at $5 \cdot 10^{-3}$ for *DVL* and $5 \cdot 10^{-2}$ for *Alex* to ensure they do not dominate the process.

Low-frequency Disruptions (Stage 2— §4.5.2). Each filter is characterized by the dimensionality $M_t \times N_t$ of $\boldsymbol{\mathcal{K}}_t^*$ and permissiveness σ_{c_t}. We find σ_{c_t}=$5 \cdot 10^{-2}$ optimal. When choosing the kernel sizes and number of filters T, resources may be prohibitive as we have a kernel per pixel, making a filter scale as the number of pixels multiplied by its kernel size. We found T=9 with different sizes defeats all studied systems. This architecture was chosen (semi-randomly) to target various potential geometries of the watermark under our computational limitations (*40GB A100 GPU*) but can be extended with better hardware. Yet, our findings indicate that this may not be necessary. The ordered sizes are $(21, 5)$, $(5, 5)$, $(17, 33)$, $(7, 7)$, $(47, 5)$, $(33, 17)$, $(17, 17)$, $(5, 5)$, and $(3, 3)$. Our spectral loss consists of *MPL* and *FRL*. While we maximize both, *FRL* is a supporting regularization term—see §4.5.2. Hence, we deem a sample optimal if it maximizes *MPL*. Yet, λ_{FRL} must account for *FRL*, ensuring the outputs meet the conditions it enforces. λ_{FRL}=5 was found to yield best results, but it can be adjusted if the two components do not meet their expected goals (see §4.5.2). We found $\lambda_{\|\|_2}$=$2.5 \cdot 10^{-4}$ works best but it can be calibrated similarly.

We universally use *DVL* (see §4.5.1). As low-frequency perturbations do not introduce sporadic peaks, the remaining parameters can be selected uniformly with little regard to the image size. The exception is the threshold t_{DVL}, as smaller images ($\leqslant$ 256×256) are more sensitive to structural changes (unlike high-frequency perturbations that these images'

low resolution masks), exhibiting visible differences for $t_{DVL} > 10^{-3}$, while larger images can withstand $3\times$ this change. With an initial $c_{DVL} = 10^6$, a single step running for 500 iterations is sufficient. We use Adam with a 10^{-2} learning rate for the filters and $2 \cdot 10^{-3}$ for δ, decayed by 0.99 if *MPL* fails improve by $5 \cdot 10^{-2}$ for 10 iterations. As *DVL*'s impact is weakened due to this procedure's structural nature, we allow its gradients to be more dominant, clipped at 1.0.

Filters operate on all image channels identically. Hence, as the reason behind including the modifier δ is the filters' ability to smoothen its otherwise high-frequency changes, δ must also affect all channels similarly. Thus, we use a *gray-scale* modifier δ that applies the same changes to all channels. While manipulating spectral amplitudes impacts the texture and results in less critical changes, phase modifications are more visible (see §4.4.1). Hence, we should ideally restrict our filters to manipulate magnitudes. The frequency response of a filter does not change the phase *iff* it is even-symmetric (identical when reflecting each of its quadrants across both axes). This holds for *basic* (convolutional) filters with an identical kernel at all locations. At the same time, our filters will converge to similar kernels for neighboring pixels that are usually of similar visual importance and can be altered similarly, indicating that each region will behave as if it were "convolved with a basic kernel". Thus, we can locally enforce that the modifications do not change the phase by enforcing this same symmetry. Accordingly, instead of optimizing all weights of each kernel, we focus only on the first quadrant of each kernel and reflect it across both axes.

B.2 Watermarking Schemes' Configuration

The (author-reported) watermark lengths are in Table B.1.

B.3 Baseline Results against Image Manipulations

The kernel size for *Gaussian (Guided) Blurring* is 5×5. We exploit *Guided Blurring*'s edge-preserving properties to increase the *std.* (to 5.0). To control the modulation at edges, these accept an additional *color kernel*'s *std.* (see §4.5.2), for which we have three values: 0.1, 1.0, and 5.0. Yet, 1.0 and 5.0 are overly destructive and do not preserve quality, chosen only to explore robustness. All parameters are in Table B.2.

Scheme	Length (# **bits**)
Yu1	128
Yu2	100
HiDDeN	30
PTW	40
StableSignature	48
StegaStamp	100
TRW	NA

Note: TRW's watermarks are not binary; thus, this metric is not applicable.

Table B.1: Watermark lengths.

Results. Most systems are robust to all *practical* manipulations. Despite resisting all other modifications, *Yu2* breaks completely under *cropping*. Hence, unlike other schemes, it highly depends on spatial attributes[1]. *Yu1, HiDDeN* and *TRW* exhibit a considerable drop in the *bit accuracies* (*inverse distance*) after *cropping*, although their overall detection is not affected. This is not surprising, as the watermarking spectral modifications must be spread throughout the image to minimize visual cues, implying that a portion may be allocated to the background. Yet, they retain robustness by restricting most of these changes to central regions.

Yu1, StegaStamp and *TRW* resist all manipulations, while *StableSignature* can withstand realistic, non-obtrusive modifications, making it also robust for practical purposes. *PTW* is slightly less powerful, losing some of its efficacy under even non-destructive operations, such as *Quantization* with a factor of 10 and *JPEG* with quality 80 that decrease its performance to 46% and 76%, respectively. *HiDDeN* exhibits similar behavior, resisting all realistic manipulations aside from *JPEG* that brings its detection to 43% even when used with a reasonably good quality of 110, or *Blurring* with *std.* = 1.0 that lowers its detection score to 5.0%.

It is expected that some *non-semantic* systems (e.g., *PTW, HiDDeN* and *StableSignature*) begin to break under spectral manipulations such as *JPEG* and *Blurring*. As explained in §4.4.1, constructing spectral *non-semantic* watermarks should be done in an input-specific manner that is also persistent, which is a challenging task that will be restricted to volatile watermarks unless the scheme is sufficiently advanced. This explains

[1]Yu et al. [281] only evaluate a single *ProGAN* [125] architecture under manipulations, whereas we chose their *StyleGAN2* [127] model.

Manipulation \ Scheme		*Yu1* (61%)		*Yu2* (63%)		*HiDDeN* (73%)		*PTW* (70%)		*StableSig.* (69%)		*StegaStamp* (63%)		*TRW* (0.0141)	
		Bit Acc↑	Detect↑	Bit Acc↑	Detect↑	Bit Acc↑	Detect↑	Bit Acc↑	Detect↑	Bit Acc↑	Detect↑	Bit Acc↑	Detect↑	Inv. Dist.↑	Detect↑
None		98.42%	100%	99.71%	100%	99.16%	100%	97.96%	100%	98.97%	100%	99.95%	100%	0.022	100%
Cropping	*ratio = 10%*	74.56%	100%	53.71%	0%	86.84%	100%	93.9%	100%	97.86%	100%	93.96%	100%	0.0155	100%
Blur	*std. = 0.5*	98.13%	100%	99.55%	100%	96.45%	100%	96.93%	100%	98.52%	100%	99.95%	100%	0.0221	100%
	std. = 1.0	94.6%	100%	99.73%	100%	62.15%	5.00%	82.25%	100%	86.65%	99%	99.88%	100%	0.0207	100%
Guided Blur	*color std. = 0.1*	87.91%	100%	99.32%	100%	75.6%	76%	94.41%	100%	96.38%	100%	99.91%	100%	0.0212	100%
	*color std. = 1.0**	85.32%	100%	99.43%	100%	56.12%	0%	78.66%	98%	71.25%	69%	99.93%	100%	0.02	100%
	*color std. = 5.0**	84.9%	100%	99.62%	100%	53.81%	0%	76.42%	92%	66.54%	50%	99.93%	100%	0.0199	100%
Quantization	*factor = 8*	85.86%	100%	99.16%	100%	89.34%	98%	74.92%	93%	98.83%	100%	99.83%	100%	0.0207	100%
	factor = 10	82.51%	100%	98.7%	100%	86.51%	98%	68.34%	46%	99.04%	100%	99.8%	100%	0.0201	100%
JPEG	*quality = 110*	89.58%	100%	99.55%	100%	70.73%	43%	87.66%	100%	97.68%	99%	99.94%	100%	0.0215	100%
	quality = 80	76.38%	100%	93.15%	100%	66.48%	16%	72.17%	76%	90.66%	100%	99.89%	100%	0.202	100%

Note: The first row shows the results in the absence of any manipulations. Numbers in parentheses are the detection thresholds, while Detect is the portion of the images meeting them. For attacks with multiple configurations, lower rows correspond to stronger distortions. ∗ denotes overly-destructive configurations.

Table B.2: Performance of watermarking schemes against basic manipulations.

why the more sophisticated *StableSignature* and *PTW* outperform *HiDDeN*. Yet, allowing for severe manipulations via stronger *(Guided) Blurring* breaks *StableSignature* as well, although this is only included to demonstrate this phenomenon that attests to the spectral nature of these watermarks and is not a valid attack (see §4.6.1). *Quantization* can cause sharp pixel shifts (high-frequency noise), explaining *PTW*'s vulnerability.

With few exceptions, which we attribute to slight mismatches in experimental settings, our findings align with those attained by the authors of the considered schemes who evaluate the same attacks with similar parameters.

B.4 Outputs of *UnMarker* vs. The *SOTA*

Fig. B.1 displays examples of watermarked images and the outputs obtained by subjecting them to advanced attacks.



Figure B.1: Outputs of the three removal attacks. The *VAEAttack* over-smooths images, resulting in them losing crucial information and occasionally looking cartoonish. Compared to the *DiffusionAttack*, *UnMarker*'s outputs better resemble the watermarked images. The *DiffusionAttack* also introduces semantic incoherences such as additional teeth in the case of *PTW* and omits identifying details such as freckles on the forehead for *StegaStamp*. *PTW*'s watermarked image itself is of inferior quality with an unnatural patch. The *DiffusionAttack* retains this artifact while *UnMarker* eliminates it, enhancing the image.

Appendix C

DiffBreak: Is Diffusion-Based Purification Robust?

C.1 Equivalence of DDPM and VP-SDE

An alternative to the *continuous-time* view described in §2.1.2 (i.e., **VP-SDE**) is *Denoising diffusion probabilistic modeling* (**DDPM**) [106, 227], which considers a *discrete-time* framework (*DM*) where the forward and reverse passes are characterized by a maximum number of steps T. Here, the forward pass is a Markov chain:

$$\boldsymbol{x}_i = \sqrt{1-\beta_i}\boldsymbol{x}_{i-1} + \sqrt{\beta_i}\boldsymbol{z}_i$$

where $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0},\ \boldsymbol{I}_d)$ and β_i is a small positive noise scheduling constant. Defining $dt = \frac{1}{T}$, we know due to Song et al. [230] that when $T \longrightarrow \infty$ (i.e, $dt \longrightarrow 0$, which is the effective case of interest), this converges to the *SDE* in *eq.* (2.1) (with the drift and diffusion function $\boldsymbol{f}$ and g described in §2.1.2). The reverse pass is also a Markov chain given as:

$$d\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{i-1} - \hat{\boldsymbol{x}}_i = \frac{1}{\sqrt{1-\beta_i}}((1-\sqrt{1-\beta_i})\hat{x}_i + \beta_i \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_i, i)) + \sqrt{\beta_i}\boldsymbol{z}_i \tag{C.1}$$

When $T \longrightarrow \infty$, $d\hat{\boldsymbol{x}}$ converges to *eq.* (2.4) (see [230]). Thus, the two views are effectively equivalent. Accordingly, we focus on the continuous view, which encompasses both frameworks. For discrete-time paradigms, $\boldsymbol{x}(t)$ effectively denotes $\boldsymbol{x}_{\lfloor \frac{t}{dt} \rfloor}$.

C.2 Details on Our DiffGrad Gradient Module

In Appendix C.2.1, we identify key implementation challenges in prior work, detailing how **DiffGrad** systematically resolves each one. Then, in Appendix C.2.2, we empirically demonstrate their impact on *DBP*'s gradients when left unaddressed, reinforcing their practical significance and validating the need for our reliable module. Appendix C.2.3 presents the pseudo-code for our memory-efficient checkpointing algorithms, offering a transparent blueprint for reproducible and correct gradient-enabled purification. Finally, in Appendix C.2.4, we verify the correctness of our implementation, confirming that the gradients computed by **DiffGrad** match those produced by a true differentiable purification pipeline.

C.2.1 The Challenges of Applying Checkpointing to *DBP* and How DiffGrad Tackles Them

High-Variance Gradients

The challenge of high-variance gradients due to insufficient EOT samples is introduced in §5.3.2. We defer its empirical analysis to Appendix C.2.2, where we quantify the gradient variability under different sample counts. Practical implications for performance and throughput—enabled by **DiffGrad**'s design—are explored in the ablation study in Appendix C.6. Accordingly, we focus below on the remaining implementation issues.

Time Consistency

Rounding issues may emerge when implementing checkpointing over *torchsde* solvers: Given the starting time t^*, the solver internally converts it into a *PyTorch* tensor, iteratively calculating the intermediate outputs by adding negative time increments to this tensor. On the other hand, checkpointing requires re-calculating the intermediate steps' samples during back-propagation. If this code is oblivious to *torchsde*'s conversion of the initial time into a *PyTorch* tensor, it will continue to treat it as a floating point number, updating it with increments of the time step to obtain each intermediate t used to re-compute the dependencies. The *PyTorch* implementation does not aim for 100% accuracy, leading to minute discrepancies in the current value of t compared to pure floating-point operations. These inaccuracies accumulate over the time horizon, potentially severely affecting the gradients. Instead, we ensure both the solver and checkpointing code use the same objects (either floating points or tensors).

Reproducing Stochasticity

CuDNN-Induced Nondeterminism. We found that even in deterministic purification pipelines, subtle nondeterminism from PyTorch's backend can cause discrepancies between forward and backward propagations. Specifically, CuDNN may select different convolution kernels during the recomputation of intermediate states in checkpointing, leading to minute numerical drift.

To eliminate this, *DiffBreak* explicitly forces CuDNN to behave deterministically, guaranteeing consistent kernel selection and numerical fidelity. While this source of error is subtle, it is measurable: for *DiffPure* with $t^* = 0.1$ and $dt = 0.001$, we observed raw and relative gradient discrepancies of $1e-4$ and $7.28e-5$ when all other issues have been addressed. With this fix, the discrepancy dropped to exactly zero. Although less severe than other sources of error (e.g., time inconsistency or missing gradient paths), this level of precision is critical in setups where gradients are small to prevent sign changes that can affect optimization trajectories. Our inspection of prior work's code reveals this drift was not previously addressed.

Noise Sampler for Guidance Robustness. While the contribution of the stochastic component of $d\hat{\boldsymbol{x}}$ often cancels analytically in vanilla **VP-SDE**-style purification (see §2.1.2), meaning it does not affect the gradient and therefore is not required to be accurately reproduced during checkpointing, this is not guaranteed in all variants. In particular, many guidance-based schemes—see §5.3.2—integrate stochastic terms within the guidance function itself. That is, the guidance function may incorporate stochasticity. When this occurs, gradients become sensitive to noise realizations, and failure to preserve them could break correctness. Yet, standard checkpointing only stores the intermediate $\hat{\boldsymbol{x}}(t)$s. Hence, $\hat{\boldsymbol{x}}(t+dt)$ will differ between the propagation phases as $d\hat{\boldsymbol{x}}$ is computed via different random variables.

We restructure the logic computing $d\hat{\boldsymbol{x}}$: We define a function $\boldsymbol{calc_dx}$ that accepts the noise as an external parameter and utilize a *Noise Sampler* $\boldsymbol{NS}$, initialized upon every forward propagation. For each t, $\boldsymbol{NS}$ returns all the necessary random noise vectors to pass to $\boldsymbol{calc_dx}$. Instead of $\hat{\boldsymbol{x}}(0)$ only, forward propagation also outputs $\boldsymbol{NS}$ and a state $\boldsymbol{S}$ containing all $\hat{\boldsymbol{x}}(t)$s. These objects are used to restore the path in backpropagation. This design also future-proofs *DiffBreak* for emerging guided or hybrid purifiers: any randomness on which the guidance depends is captured by $\boldsymbol{NS}$ and replayed exactly during differentiation.

The memory cost of our design is $\mathcal{O}(\frac{t^*}{|dt|})$*, storing only* $\boldsymbol{NS}$*, and all* $\hat{\boldsymbol{x}}(t)$*s, each with a*

negligible footprint ($\mathcal{O}(1)$) compared to graph dependencies [124].

Guidance Gradients

In schemes that involve guidance, it is typically obtained by applying a function $\boldsymbol{g_{fn}}$ to $\hat{\boldsymbol{x}}(t)$ at each step (note that $\boldsymbol{g_{fn}}$ may involve stochasticity—see Appendix C.2.1—as is often the case in *GDMP* [250]). For instance, in *Guided*-**DDPM** [250], the original sample $\boldsymbol{x}$ is used to influence each step of the reverse procedure to retain key semantic information, allowing for an increased budget t^* to better counteract adversarial perturbations. Effectively, it modifies *eq.* (C.1) describing the reverse pass of **DDPM** as:

$$d\hat{\boldsymbol{x}} = \frac{1}{\sqrt{1-\beta_i}}((1-\sqrt{1-\beta_i})\hat{\boldsymbol{x}}_i + \beta_i \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_i, i)) - s\beta_i \nabla_{\hat{\boldsymbol{x}}_i} \boldsymbol{GC}(\hat{\boldsymbol{x}}_i,\ \boldsymbol{x}) + \sqrt{\beta_i}\boldsymbol{z}_i$$

where $\boldsymbol{GC}$ is a guidance condition (typically a distance metric) each step minimizes by moving in the opposite direction of its gradient, while the scale s controls the guidance's influence.

That is, in the specific case of *Guided*-**DDPM**, $\boldsymbol{g_{fn}}(\hat{\boldsymbol{x}}(t)) \equiv -\nabla_{\hat{\boldsymbol{x}}(t)} \boldsymbol{GC}(\hat{\boldsymbol{x}}(t),\ \boldsymbol{x})$, where $\boldsymbol{GC}$ is a distance metric. Nonetheless, other choices for $\boldsymbol{g_{fn}}$ may be employed in general. Typically, as the goal of the guidance is to ensure key information from the original sample $\boldsymbol{x}$ is retained, $\boldsymbol{g_fn}$ will also *directly* involve this $\boldsymbol{x}$ in addition to $\hat{\boldsymbol{x}}(t)$ (e.g., $\boldsymbol{GC}$ above). Yet, a naive implementation would back-propagate the gradients to $\boldsymbol{x}$ by only considering the path through $\hat{\boldsymbol{x}}(t)$. Yet, when $\boldsymbol{g_fn}$ relies on a $\boldsymbol{guide}$ constructed from $\boldsymbol{x}$ to influence $\hat{\boldsymbol{x}}(t)$ (e.g., $\boldsymbol{guide} \equiv \boldsymbol{x}$ in *Guided*-**DDPM**), it creates additional paths from $\boldsymbol{x}$ to the loss through this $\boldsymbol{guide}$ at each step t. Accordingly, **DiffGrad** augments the process to include the gradients due to these paths. In the general case, this $\boldsymbol{guide}$ may not be identical to $\boldsymbol{x}$ but can rather be computed based on $\boldsymbol{x}$ or even completely independent (in which case no guidance gradients are collected). **DiffGrad** captures this nuance through an abstract function $\boldsymbol{g_aux}$ that, given $\boldsymbol{x}$, outputs the $\boldsymbol{guide}$. Similar to *eq.* (5.2), we have that for each t, the gradient of any function F applied to $\hat{\boldsymbol{x}}(0)$ w.r.t. $\boldsymbol{guide}$ due to the path from $\hat{\boldsymbol{x}}(t+dt)$ to $\hat{\boldsymbol{x}}(0)$ is given by:

$$\nabla^t_{\boldsymbol{guide}} F = \nabla_{\boldsymbol{guide}} \langle \hat{\boldsymbol{x}}(t+dt), \nabla_{\hat{\boldsymbol{x}}(t+dt)} F \rangle \tag{C.2}$$

as $\hat{\boldsymbol{x}}(t+dt)$ is a function of $\boldsymbol{guide}$. Recall that we are interested in F, which is the loss function over the classifier's output on $\hat{\boldsymbol{x}}(0)$. The gradient of F w.r.t. $\boldsymbol{guide}$ is a superposition of all these paths' gradients, given as:

$$\nabla_{\boldsymbol{guide}} F = \sum_t \nabla^t_{\boldsymbol{guide}} F \tag{C.3}$$

By the chain rule, F's gradient w.r.t. $\boldsymbol{x}$ due to the guidance paths, which we denote as $\nabla^g_{\boldsymbol{x}} F$, is:

$$\nabla^g_{\boldsymbol{x}} F = \nabla_{\boldsymbol{x}} \langle \boldsymbol{guide}, \nabla_{\boldsymbol{guide}} F \rangle \tag{C.4}$$

As a convention, for an unguided process or when $\boldsymbol{guide}$ and $\boldsymbol{x}$ are not related, we define the gradient returned from *eq.* (C.4) as $\nabla^g_{\boldsymbol{x}} F \equiv \mathbf{0}$, preserving correctness in general. Since $\boldsymbol{x}$ in this guided scenario traverses both the guidance and standard purification paths, the final gradient is the sum of both components.

Finally, automatic differentiation engines, by default, generate gradients without retaining dependencies on the inputs that produced them. Thus, when the guidance itself is in the form of a gradient as in the case of *Guided*-**DDPM** or other potential alternatives, its effects will not be back-propagated to $\boldsymbol{x}$ through any of the two paths described above, despite our proposed extensions. **DiffGrad** alters this behavior, retaining the dependencies of such gradient-based guidance metrics as well.

Proper Use of The Surrogate Method [148]

As noted in §5.3.2, **DiffHammer** [251] adopts the **surrogate** method (originally proposed by Lee and Kim [148]) for gradient computation, as acknowledged in their Appendix C.1.1. This approach approximates *DBP*'s gradients by replacing the standard fine-grained reverse process implementations (e.g., $dt = 0.001$ is typically used for *DM*s as mentioned in §2.1.2) with a coarser one (e.g., $\bar{dt} = 0.01$), thereby enabling memory-efficient differentiation using standard autodiff. For instance, when using a diffusion time $t^* = 0.1$ for *DBP* (see §5.2), the fine-grained process would require 100 steps, which is computationally infeasible (memory-exhaustive) for gradient computation without checkpointing. In contrast, the coarse-grained process would only require 10 steps, which is possible.

Specifically, the forward pass (diffusion) is computed using the closed-form solution from *eq.* (2.2), while the reverse pass uses $\bar{dt}$, enabling efficient gradient computation. Notably, this replacement only occurs for the purpose of updating the attack sample (i.e., computing the gradients) and happens during both forward and backpropagation so that the two phases match. In contrast, when evaluating the generated *AE*, it is purified via the standard fine-grained process that uses dt, as this is the true (full) procedure the model owner (defender) would execute. Although the surrogate process yields approximate gradients for a slightly different process, Lee and Kim [148] found it effective and more accurate than other approximations like the **adjoint** method [155] (despite remaining suboptimal—see Appendix C.4).

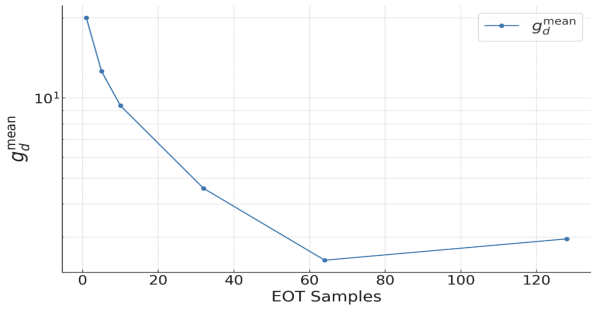
However, a manual inspection of **DiffHammer**'s code[1] reveals that their implementation of this surrogate method contains a critical issue. Rather than computing the reverse process with the coarse $\bar{dt}$ as intended for attack optimization during both forward and backpropagation, they first run the reverse pass during forward propagation with the standard dt, storing intermediate states. Then, during backpropagation, they attempt to backpropagate gradients using checkpoints spaced at $\bar{dt}$ intervals, but instead reuse the stored dt-based forward states. This leads to a mismatch: gradients intended for steps of size $\bar{dt}$ are applied to states computed with dt, effectively disconnecting the computation graph and introducing significant gradient errors. In practice, this behaves like a hybrid between checkpointing and **BPDA** [20], where gradients are approximated around fixed anchor points without correctly tracking the reverse dynamics.

As we show in §5.4.2, this implementation leads to significant attack performance degradation compared to the full gradients. It illustrates the fragility of gradient computation in diffusion attacks and further motivates the need for our reliable **DiffGrad** module.

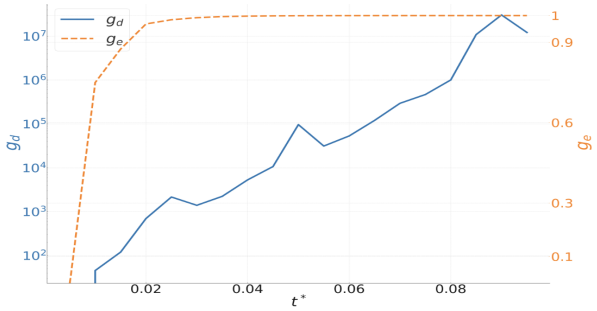
C.2.2 Demonstrating The Effects of Incorrect Backpropagation

We empirically demonstrate the impact of the identified issues on *DBP*'s gradients in Fig. C.1 and briefly explain each result below. As numerical results for CuDNN-induced nondeterminism (i.e., unhandled stochasticity) are already presented in Appendix C.2.1, we focus here on the remaining four issues detailed in Appendix C.2.1:

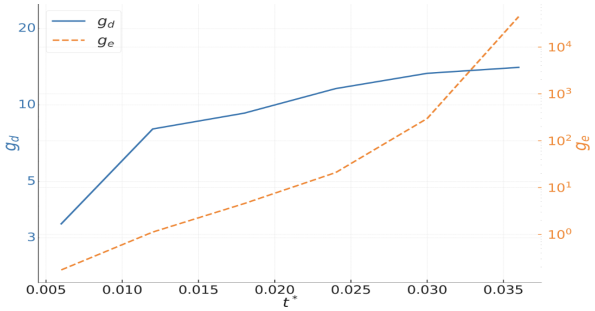
[1] https://github.com/Ka1b0/DiffHammer (accessed in May 2025)



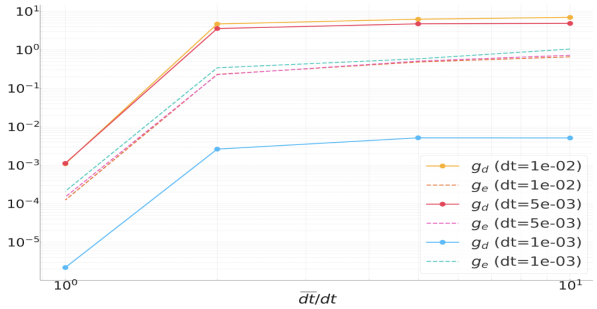
(a) Effect of insufficient EOT samples



(b) Effect of time inconsistency



(c) Effect of ignoring guidance gradients



(d) Effect of incorrect surrogate gradients

Figure C.1: Effects of the identified issues in *DBP*'s backpropagation. Each subfigure visualizes a specific error source.

(a) Effect of insufficient EOT samples. Fig.C.1a illustrates the impact of using too few EOT samples when computing gradients—see§5.3.2 and Appendix C.2.1. While prior works [251, 124, 148, 162, 182] typically use 10–20 samples, such low counts, while somewhat effective, may introduce significant variance, resulting in noisy gradients and suboptimal attacks.

To quantify this, we run the following experiment on a ***CIFAR-10*** sample $\boldsymbol{x}$ purified with *DiffPure* (t^*=0.1, dt=10^{-3}). For each EOT count N (shown on the x-axis), we generate N purified copies, compute the average loss, and backpropagate to obtain the EOT gradient. This is repeated 20 times to yield 20 gradients $g_1^N, \ldots, g_{20}^N$ for each N. We then define:

$$g_d^{mean}(N) = \frac{1}{20} \sum_{i=1}^{20} \underset{j}{max} \, \|\boldsymbol{g}_i^N - \boldsymbol{g}_j^N\|_2$$

This metric reflects the worst-case deviation between gradients under repeated EOT

sampling. By the central limit theorem (see §5.3.1), the variance of EOT gradients decays with N, and thus $g_d^{\text{mean}}(N) \to 0$ as $N \to \infty$. The curve in Fig. C.1a captures this decay and lets us identify a threshold where variance becomes acceptably low.

The plot confirms that gradient variance decreases sharply with the number of EOT samples. While $N = 10$—the typical choice in prior work—is somewhat effective and viable under resource constraints, it remains suboptimal. We observe that variance continues to drop until around $N = 64$, after which it plateaus. This suggests that although larger N values like 128 yield marginally better stability, the bulk of the benefit is realized by $N = 32$–64. Thus, $N = 10$ is a practical baseline, but $N \geqslant 64$ should be preferred when accuracy is critical and compute allows, corroborating our claims in §5.3.2.

(b) Effect of time inconsistency. As detailed in §5.3.2 and Appendix C.2.1, rounding inconsistencies can emerge when using checkpointing with *torchsde* solvers. These stem from how the solver internally handles time as a *PyTorch* tensor, which may diverge subtly from floating-point representations if not carefully synchronized. While such discrepancies are negligible for short purification paths (small t^*), they accumulate significantly over longer trajectories—particularly at standard *DBP* settings like t^*=0.1 with dt=10^{-3} (i.e., 100 reverse steps), as in *DiffPure* on ***CIFAR-10***.

To quantify this effect, we purify a fixed ***CIFAR-10*** sample $\boldsymbol{x}$ using values of t^* from 0.005 to 0.1 in steps of 0.005. For each t^*, we compute gradients using **DiffGrad** under two conditions: (1) without correcting the inconsistency (yielding $\boldsymbol{g}_{nf}$), and (2) with the issue fixed (yielding $\boldsymbol{g}_f$), both using the same stochastic path. We then compute two metrics: (1) $g_d = \|\boldsymbol{g}_f - \boldsymbol{g}_{nf}\|_2$ (absolute error), and (2) $g_e = \frac{\|\boldsymbol{g}_f - \boldsymbol{g}_{nf}\|_2}{\|\boldsymbol{g}_f\|_2}$ (relative error). These quantify the divergence introduced by uncorrected rounding—both in magnitude and in proportion to the true gradient—and illustrate its increasing severity with longer reverse trajectories.

Fig. C.1b confirms that even minute rounding discrepancies in time handling, if uncorrected, can completely corrupt gradients over longer diffusion trajectories. Both the absolute error g_d (blue) and the relative error g_e (orange) grow rapidly with t^*. By $t^* \geqslant 0.03$, g_e exceeds 90%, and for the typical setting of $t^* = 0.1$, the relative error reaches 100%, indicating that the gradients are potentially nearly orthogonal to the correct direction. This validates our claim that unpatched checkpointing introduces severe errors and demonstrates that our **DiffGrad** fix is essential for correct gradient computation in practical *DBP* setups.

(c) Effect of ignored guidance gradients. We evaluate the impact of neglecting guidance gradients, a critical problem in current attacks against *Guided DBP* defenses (e.g.,

GDMP, *MimicDiffusion*)—see §5.3.2 and Appendix C.2.1. As explained earlier, guidance introduces additional gradient paths from the input $\boldsymbol{x}$ to $\hat{\boldsymbol{x}}(t)$, typically requiring second-order derivatives to capture. Existing approaches ignore these paths, leading to incomplete gradients and suboptimal attacks. To our knowledge, **DiffGrad** is the first system to correctly handle them.

To quantify this effect, we purify a fixed ***CIFAR-10*** sample $\boldsymbol{x}$ using *GDMP* (setup in Appendix C.3) while varying the purification horizon $t^* \in \{0.006, 0.012, \ldots, 0.036\}$. For each t^*, we compute gradients using **DiffGrad** with and without accounting for guidance—yielding $\boldsymbol{g}_f$ and $\boldsymbol{g}_{nf}$, respectively—under the same stochastic path. We then compute the absolute and relative gradient error (g_d, g_e) as in previous experiments.

The findings in Fig. C.1c clearly demonstrate the severe impact of neglecting guidance gradients in *Guided DBP*. Both the raw error (g_d) and the relative error (g_e) increase steeply with t^*, confirming that as the purification path grows longer, the omitted gradient paths—stemming from guidance dependencies—dominate the overall gradient signal. Most notably, g_e exceeds 10^4 at $t^* = 0.036$, indicating that the gradient used in prior works is essentially meaningless for optimization, as it no longer necessarily aligns with the true direction of steepest descent.

These findings directly explain the poor attack success rates previously observed on guided defenses like *GDMP* and *MimicDiffusion*, and reinforce why our attacks, which correctly account for these guidance gradients via **DiffGrad**, achieve drastically superior performance—see §5.4.2 and §5.6. In short, the guidance mechanism, when improperly handled, not only fails to help but potentially worsens attack effectiveness by yielding incorrect gradient signals.

(d) Effect of incorrect surrogate gradients. As explained in §5.3.2 and Appendix C.2.1, prior work [251] uses the **surrogate** method [148] to enable efficient gradient computation for *DBP*. However, we identify a critical divergence in **DiffHammer**'s [251] implementation of this method that substantially compromises attack performance. This not only leads to inflated robustness estimates for *DBP*, but also to misleading conclusions about the relative effectiveness of the standard gradient-based attacks versus **DiffHammer**'s proposed enhancements—see §5.4.2. Specifically, as detailed in Appendix C.2.1, the implementation of Wang et al. [251] mismatches the time discretization used in the forward and backward passes: the forward pass computes intermediate denoised states $\hat{\boldsymbol{x}}(t)$ using a fine step size dt, while the backward pass attempts to propagate gradients at a coarser granularity $\bar{dt} > dt$ using those same (misaligned) forward states. This inconsistency leads to gradients being applied to states generated under a different dynamics, corrupting the

computation and severely degrading the attack.

To validate this, we conduct a final experiment: For three different values of $dt \in \{0.001,\ 0.005,\ 0.01\}$ and $t^* = 0.1$, we purify a fixed ***CIFAR-10*** sample $\boldsymbol{x}$. For each dt, we evaluate four surrogate variants using $\bar{dt} \in \{dt,\ 2dt,\ 5dt,\ 10dt\}$. For every configuration, we compute $\boldsymbol{g}_{nf}$ using Wang et al. [251]'s surrogate implementation and $\boldsymbol{g}_f$ using the correct implementation. As before, we report both g_d and g_e to quantify the absolute and relative gradient error due to the mismatch between dt and $\bar{dt}$. Each experiment is repeated 10 times, and we report the mean values.

The results in Fig. C.1d clearly validate our claim: the gradient errors induced by Wang et al. [251]'s surrogate implementation grow substantially with increasing $\bar{dt}/dt$, saturating quickly even for modest mismatches. Both absolute error g_d and relative error g_e consistently worsen across all dt configurations, indicating that gradients are being applied to the wrong points in the computation graph.

Crucially, Wang et al. [251] use $dt = 0.01$ and $\bar{dt} = 2dt = 0.02$, a setup which already produces substantial degradation ($g_d \approx 3$, $g_e \approx 1$), confirming that their reported results rely on gradients that diverge significantly from the intended ones. These errors propagate into the attack, weakening its effectiveness and misleadingly suggesting that first-order attacks are inherently inferior. This experiment conclusively demonstrates that Wang et al. [251]'s surrogate misuse is not a minor detail, but a critical bug that undermines their central claims.

We note that **DiffGrad** also provides a correct implementation of the **surrogate** method (despite the full correct gradient being the main method considered in Chapter 5).

C.2.3 Pseudo-Code for Our Memory-Efficient Gradient-Enabled Purification with DiffGrad

Forward Propagation. **DiffGrad**'s forward propagation logic is in Algorithm C.1. The code in blue is optional, pertaining to the use of guidance. We highlight in red the portions that differ from standard forward propagation. First, we generate the guidance ***guide*** from $\boldsymbol{x}$ (line 1) and disable all graph dependency storage (line 2), enabling our code to run efficiently without attempting to store graphs that would lead to memory failures. Afterward (lines 3-5), we initialize $\boldsymbol{S}$ as an empty list and draw a random seed that is then used to invoke the abstract function ***init_noise_sampler***, which returns a noise sampler that provides a reproducible random path for the backpropagation phase (see §5.3.2). After the input is diffused (lines 6-8) via *eq.* (2.2), lines 9-15 correspond to the

Algorithm C.1 Differentiable Purification with ***DiffGrad*** — Forward Propagation

Require: Sample $\boldsymbol{x}$, Score model $\boldsymbol{s}_\theta$, Optimal diffusion time t^*, step size dt, Noise scheduler β, Reverse diffusion function **calc_dx**, Noise sampler initializer **init_noise_sampler**, Guidance condition $\mathbf{g_{fn}}$, Guidance scale s, Auxiliary guidance extractor **g_aux**

$steps \leftarrow \left|\frac{t^*}{dt}\right|$, **guide** $\leftarrow$ **g_aux**$(\boldsymbol{x})$ /* Calc. #steps and init. guide */

disable_dependencies() /* Dependencies enabled during forward prop. Disable them. */

S $\leftarrow$ [] /* Saved state (will eventually hold all intermediate reverse steps' outputs). */

$seed \leftarrow$ **random_seed**() /* Seed to initialize noise path */

NS $\leftarrow$ **init_noise_sampler**$(seed)$ /* **Reproducible** sampler */

$\alpha \leftarrow$ **calc_alpha**(β) /* Calculate α factors from *eq.* (2.2) */

Draw $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$

$\hat{\boldsymbol{x}} \leftarrow \sqrt{\alpha(t^*)}\boldsymbol{x} + \sqrt{1-\alpha(t^*)}\boldsymbol{\epsilon}$ /* Diffuse according to *eq.* (2.2) */

for $i \leftarrow steps, steps-1, \ldots, 1$ **do**

  S.append$(\hat{\boldsymbol{x}})$ /* Set $\mathbf{S}[i] = \hat{\boldsymbol{x}}(t)$. */

  step_noise $\leftarrow$ **NS.sample**(i) /* Sample the random noise used to calculate $d\hat{\boldsymbol{x}}$ at step i. */

  $d\hat{\boldsymbol{x}} \leftarrow$ **calc_dx**$(\hat{\boldsymbol{x}}, \boldsymbol{s}_\theta, i, dt, \beta,$ /* Calc. $d\hat{\boldsymbol{x}}$ according to

    step_noise$, \mathbf{g_{fn}}, s,$ **guide**$)$ *eq.* (2.4) */

  $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} + d\hat{\boldsymbol{x}}$ /* Update $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(t+dt)$. */

end for

enable_dependencies() /* Re-enable dependencies. */

return $\hat{\boldsymbol{x}}$, **S**, **NS**, **guide**

reverse pass: At each step t (effectively i), $\hat{\boldsymbol{x}}$ (that now represents $\hat{\boldsymbol{x}}(t)$) is first appended to $\boldsymbol{S}$, which will eventually contain all such intermediate outputs (line 10). The noise provided by $\boldsymbol{NS}$ for the current step i is then retrieved (line 11) and used to compute $d\hat{\boldsymbol{x}}$ (line 12). $d\hat{\boldsymbol{x}}$ is then added to $\hat{\boldsymbol{x}}$ so that its current value becomes $\hat{\boldsymbol{x}}(t+dt)$. This repeats until $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(0)$. Unlike the naive implementation, we only store the intermediate results. For efficiency, we also avoid saving the random noise for each step i but utilize $\boldsymbol{NS}$ to reproduce those variables on demand. Before termination, we re-enable dependency storage (line 16) to ensure our code does not interfere with the execution of any other modules. Finally, $\hat{\boldsymbol{x}}(0)$ is returned together with the state $\boldsymbol{S}$ and the sampler $\boldsymbol{NS}$, which are stored internally for reproducibility during backpropagation.

DBP Parallelism. Although Algorithm C.1 is presented for a single purification path, **DiffGrad** natively supports parallel purification of N stochastic copies to potentially obtain a significant speedup when computing higher-quality EOT gradients—see §5.3.2. For simplicity, we abstracted away the batch logic in the pseudo-code. In practice, prior to line 1, the input $\boldsymbol{x}$ is replicated N times (N is an additional argument that can be provided to Algorithm C.1). Line 4 returns N random seeds, and the resulting sampler **NS** manages N reproducible random paths. Line 7 draws N distinct noise samples to generate N diffused versions of $\boldsymbol{x}$, and all subsequent steps operate copy-wise in parallel across these N instances.

Backpropagation. **DiffGrad**'s backpropagation logic is in Algorithm C.2. Similar to earlier, red text refers to operations that deviate from traditional backpropagation, while blue lines are optional (guidance-related). In addition to the usual gradient $\boldsymbol{grad}$ w.r.t. $\hat{\boldsymbol{x}}(0)$, the inputs include multiple parameters normally exclusive to forward propagation, as they are required to re-calculate the dependencies. Additionally, the algorithm accepts the saved state $\boldsymbol{S}$, and the same noise sampler $\boldsymbol{NS}$ to retrieve the stochastic path of the forward propagation. Before providing details, we note that by definition $\forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^d$, it holds that:

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_d \boldsymbol{A} \odot \boldsymbol{B}$$

where $\odot$ denotes the element-wise product. Therefore, in order to calculate the gradients w.r.t. $\hat{\boldsymbol{x}}(t)$ and $\boldsymbol{guide}$ as described in *eq.* (5.2) and *eq.* (C.2), we may define an objective at each step t as:

$$Obj_t = \sum_d (\hat{\boldsymbol{x}}(t+dt) \odot \nabla_{\hat{\boldsymbol{x}}(t+dt)} F) \tag{C.5}$$

and take its gradient w.r.t. the elements of interest, which explains the steps in our pseudo-code in Algorithm C.2.

Algorithm C.2 Differentiable Purification with ***DiffGrad*** — Backpropagation

Require: Loss gradient $\boldsymbol{grad}$ w.r.t. $\hat{\boldsymbol{x}}_0$, Sample $\boldsymbol{x}$, Score model $\boldsymbol{s}_\theta$, Optimal diffusion time t^*, step size dt, Noise scheduler β, Reverse diffusion function **calc_dx**, State $\mathbf{S} = \{\hat{\boldsymbol{x}}(dt * i) | i \in [\![|\frac{t^*}{dt}|]\!]\}$, Noise sampler **NS**, Auxiliary guidance input **guide**, Guidance function $\mathbf{g_{fn}}$, Guidance scale s

1: $steps \leftarrow \left|\frac{t^*}{dt}\right|$
2: **g_grad** $\leftarrow \mathbf{0}$ /* Init. gradient w.r.t. guidance input */
3: **for** $i \leftarrow 1, 2, \ldots, steps$ **do**
4: $\quad \hat{\boldsymbol{x}} \leftarrow \mathbf{S}[i]$ /* Set $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(t)$ */
5: $\quad$ **step_noise** $\leftarrow$ **NS.sample**(i) /* Retrieve noise for step i */
6: $\quad$ **enable_dependencies**()
7: $\quad d\hat{\boldsymbol{x}} \leftarrow$ **calc_dx**$(\hat{\boldsymbol{x}}, \boldsymbol{s}_\theta, i, dt, \beta,$
8: $\qquad$ **step_noise**, $\mathbf{g_{fn}}, s,$ **guide**$)$
9: $\quad \hat{\boldsymbol{x}}_{+dt} \leftarrow \hat{\boldsymbol{x}} + d\hat{\boldsymbol{x}}$ /* $\hat{\boldsymbol{x}}_{+dt} = \hat{\boldsymbol{x}}(t + dt)$ */
10: $\quad Obj_t \leftarrow \sum(\hat{\boldsymbol{x}}_{+dt} \odot \boldsymbol{grad})$ /* Objective due to *eq.* (C.5) */
11: $\quad$ **disable_dependencies**()
12: $\quad \boldsymbol{grad} \leftarrow \nabla_{\hat{\boldsymbol{x}}} Obj_t$ /* Update gradient w.r.t. $\hat{\boldsymbol{x}}(t)$ (*eq.* (5.2)) */
13: $\quad$ **g_grad** $\leftarrow$ **g_grad** $+ \nabla_{\mathbf{guide}} Obj_t$ /* Update **guide** gradient (*eq.* (C.3)) */
14: **end for**
15: $\alpha \leftarrow$ **calc_alpha**(β)
16: $\boldsymbol{grad} \leftarrow \boldsymbol{grad} * \sqrt{\alpha(t^*)}$ /* Loss gradient w.r.t. $\boldsymbol{x}$ (*eq.* (2.2)) */
17: **g_grad** $\leftarrow \nabla_{\boldsymbol{x}} \sum($**guide** $\odot$ **g_grad**$)$ /* Guidance gradient w.r.t. $\boldsymbol{x}$ (*eq.* (C.4)) */
18: $\boldsymbol{grad} \leftarrow \boldsymbol{grad} +$ **g_grad** /* Merge loss and guidance gradients */
19: **return** $\boldsymbol{grad}$

The procedure begins by creating a variable $\boldsymbol{g_grad}$ and setting it to $\mathbf{0}$ (line 2). This will later be used to store the guidance gradients (see Appendix C.2.1). For each time step t (i.e., step i), starting from $t' = -dt$ ($i = 1$), the process (lines 3-14) first retrieves $\hat{\boldsymbol{x}}(t)$ from the saved state S (line 4) and the corresponding random noise for that step used during forward propagation (line 5) and computes $\hat{\boldsymbol{x}}(t + dt)$, denoted as $\hat{\boldsymbol{x}}_{+dt}$ (lines 7-9). Importantly, these computations are performed while storing graph dependencies (enabled on line 6 and re-disabled on line 11 to restore the normal execution state). Specifically, during the first step, we calculate $\hat{\boldsymbol{x}}(0)$ from $\hat{\boldsymbol{x}}(-dt)$. Afterward, we compute the objective Obj_t (line 10) following *eq.* (C.5) that allows us to back-propagate the gradient from $\hat{\boldsymbol{x}}(0)$ to $\hat{\boldsymbol{x}}(-dt)$ and $\boldsymbol{guide}$ using the stored dependencies, as per *eq.* (5.2) and *eq.* (C.2). $\boldsymbol{grad}$

is then updated to hold the gradient of the loss function w.r.t. $\hat{\boldsymbol{x}}(-dt)$ as desired (line 12), and the gradient of ***guide*** due to this guidance path (i.e., from ***guide*** to the loss due to ***guide*** participating directly in the calculation of $\hat{\boldsymbol{x}}(t+dt)$— see Appendix C.2.1) is added to ***g_grad*** (line 13). This process repeats until ***grad*** finally holds the gradient w.r.t. $\hat{\boldsymbol{x}}(t^*)$ and ***g_grad*** holds the sum of gradients due to all guidance paths w.r.t. ***guide*** (*eq.* (C.3)). Note that after the required gradients w.r.t. $\hat{\boldsymbol{x}}(t)$ and ***guide*** are obtained at each step, the dependencies are no longer needed and can be discarded. This is where our approach differs from traditional backpropagation algorithms, enabling memory-efficient gradient calculations (at the cost of an additional forward propagation in total). At this point (line 14), we have the gradient $\nabla_{\hat{\boldsymbol{x}}(t^*)}F$ and all is required is to use it to calculate $\nabla_{\boldsymbol{x}}F$, which is trivial due to the chain rule since the closed-form solution for $\hat{\boldsymbol{x}}(t^*) \equiv \boldsymbol{x}(t^*)$ from *eq.* (2.2) indicates that this is equivalent to $\nabla_{\boldsymbol{x}}F = \sqrt{\alpha(t^*)} * (\nabla_{\boldsymbol{x}(t^*)}F)$ as we compute on line 16. We then calculate the guidance paths' gradient w.r.t. $\boldsymbol{x}$ following *eq.* (C.4) on line 17. Finally, we sum both components, returning the precise full gradient w.r.t. $\boldsymbol{x}$.

DBP Parallelism. Algorithm C.2 is written for a single purified copy but in practice operates over a batch of N stochastic instances like Algorithm C.1. As the forward pass stores N trajectories, the backward pass propagates gradients independently for each. The operations in the pseudo-code are thus applied copy-wise in parallel across all N instances. Finally, after line 18, the N gradients are averaged, yielding the EOT gradient.

C.2.4 Verifying The Correctness of DiffGrad

To ensure the correctness of **DiffGrad**, all that is required is to verify that each $\hat{\boldsymbol{x}}(t+dt)$ computed during backpropagation (line 9 in Algorithm C.2) exactly matches the corresponding forward-computed $\hat{\boldsymbol{x}}(t+dt)$ (line 14 in Algorithm C.1). This equality guarantees that the reconstructed computation graph faithfully mirrors the one produced by standard autodiff engines during their normal (non-checkpointed) operation, ensuring exact gradient recovery since we use them to perform the necessary backpropagation between each $\hat{\boldsymbol{x}}(t+dt)$ and $\hat{\boldsymbol{x}}(t)$.

Since our forward pass explicitly stores all intermediate states in $\boldsymbol{S}$, we compare each recomputed $\hat{\boldsymbol{x}}(t+dt)$ with its forward counterpart during backpropagation. An exact match confirms that the system is computing precise gradients. We manually validated this for all timesteps.

For guidance gradients, correctness follows from the derivations in Appendix C.2.1. Once incorporated, the same matching procedure ensures their correctness as well.

C.3 Additional Details on Systems & Models

WideResNet-28-10 and WideResNet-70-16 classifiers [284] are used for ***CIFAR-10***, and ResNet-50 [100], WideResNet-50-2, and DeiT-S [71] for ***ImageNet***, similar to [182, 124]. For **VP-SDE** *DBP* (*DiffPure*) [182], the *DM*s [63, 230] are those from the original work. We also experiment with the *Guided*-**DDPM** (see §5.3.2), *GDMP* [250], due to its *SOTA* robustness, using the author-evaluated *DM*s [63, 106]. The settings match the original setup [250, 182]: For *Diffpure*, t^*=0.1 for ***CIFAR-10*** and t^*=0.15 for ***ImageNet***. For *GDMP*, a ***CIFAR-10*** sample is purified $m=4$ times, with each iteration running for 36 steps (t^*=0.036) using *MSE* guidance [250]. ***ImageNet*** uses 45 steps (m=1) under **DDPM**-acceleration [63] with *SSIM* guidance [258].

C.4 One-Shot *DBP* Baseline Comparisons Against DiffGrad's Accurate Gradients

As noted in §5.3, *DBP*'s robustness stems from two sources: inaccurate gradients and improper evaluation. As our work offers enhancements on both fronts, we evaluate each factor separately. Here, we isolate the gradient issue by re-running prior experiments under the same one-shot evaluation protocol, but with accurate gradients via our **DiffGrad** module and compare the results to those from the literature. We restrict our attacks to 10 optimization steps (with *AA*-ℓ_∞), while prior works often use up to 100, giving them a clear advantage. We evaluate on ***CIFAR-10*** with WideResNet-28-10, using $N = 128$ EOT samples as justified in §5.3.2. For papers reporting several results, we chose their best.

Baselines. To demonstrate the efficacy of exact full gradients, we compare our **DiffGrad** module against prior gradient approaches. For *DiffPure*, the **adjoint** method was originally used [182], while *GDMP* was initially evaluated using **BPDA** [20] and a **blind** variant (ignoring the defense entirely) in Wang et al. [250]. These approximations were later criticized for poor attack performance [251, 162, 124, 148]. More recently, Lee and Kim [148] proposed the **surrogate** process to approximate the gradients, performing the reverse pass with fewer steps during *AE* optimization to reduce memory usage, enabling approximate gradients via standard autodiff tools. **DiffAttack** [124], **DiffHammer** [251], and Liu et al. [162] proposed checkpointing for memory-efficient full-gradient backpropagation. However, Wang et al. [251] avoid the standard one-shot evaluation (1-evaluation) protocol and report no results under it, while Kang et al. [124] do not evaluate *GDMP* and continue to use the

adjoint method for *DiffPure*. We focus here on existing results under the 1-evaluation protocol and defer the analysis of latter works' conceptual issues (see §5.1) to §5.4.2.

Metric. Following prior work, we report *robust accuracy* (*Rob-Acc*): the fraction of samples correctly classified after the attack completes and the final adversarial example is purified and evaluated (once).

Results. Table C.1 shows our comparison. All methods achieve similar clean accuracy (Cl-Acc) without attacks, with minor variation due to sample selection. Thus, robust accuracy (Rob-Acc) differences reflect the effect of gradient methods. **Full** denotes the standard *AA* attack that uses the full exact gradients (i.e., via checkpointing).

Our approach significantly outperforms gradient approximations such as **Adjoint**, **Blind**, and **BPDA**, which is consistent with their known shortcomings. More notably, despite using only 10 optimization steps, our method reduces *DiffPure*'s Rob-Acc by 14.06% compared to Liu et al. [162], who also use exact gradients, confirming their backpropagation mismatches (see §5.3.2).

Models	Pur.	Gradient Method	Cl-Acc %	Rob-Acc %
WideResNet-28-10	*DiffPure* [182]	**Adjoint** (Nie et al. [182])	89.02	70.64
		DiffAttack (Kang et al. [124])	89.02	46.88
		Surrogate (Lee and Kim [148])†	90.07	48.28
		Full (Liu et al. [162])	89.26	62.11
		Full-DiffGrad (Ours)	89.46	**48.05**
	GDMP [250]	**Blind** (Wang et al. [250])	93.50	90.06
		BPDA (Lee and Kim [148])	89.96	75.59
		Surrogate (Lee and Kim [148])†	89.96	24.53
		Full-DiffGrad (Ours)	93.36	**19.53**

Table C.1: One-shot AA-ℓ_∞ comparison on ***CIFAR-10*** (ϵ_∞=8/255). † indicates strategy is *PGD*.

While **DiffAttack** remains slightly stronger, the difference is extremely small (1.4%) and likely stems from differing evaluation samples. Importantly, the original gap reported in Kang et al. [124] between **DiffAttack** and the standard AA-ℓ_∞ dropped by 23.76%, undermining its claimed advantage due to the per-step deviated reconstruction losses it employs and aligning with our theoretical findings in §5.3.1. As **DiffAttack** involves broader architectural changes beyond gradient logic, we provide detailed comparisons in §5.4.2, where we clearly showcase its inferiority.

Our method also slightly outperforms **Surrogate** for *DiffPure* (0.23%), but we caution against overinterpreting this small gap: under the improper one-shot evaluation, several purification paths can still cause misclassification, even if the majority yield correct labels. As such, good approximation methods like **Surrogate** may appear closer in performance

than they truly are. To verify the superiority of our **DiffGrad**'s full gradients over the **surrogate** approximation against *DiffPure*, we thus further compare the two under the realistic *MV* protocol studied in §5.4.2, executing *AA*-ℓ_∞ with the **surrogate** process to obtain the gradients when attacking the same ***CIFAR-10*** classifier considered in §5.4.2 (i.e., WideResNet-70-16) and using the same number of samples ($N = 10$) over which the majority vote is taken. We find that the **surrogate** process brings MV.Rob to 43.75% only, whereas our **Full-DiffGrad** lowers this number to 39.45%, achieving a considerable improvement of 4.3%, and unequivocally proving the advantage of exact gradient computation. Finally, for *GDMP*, our method outperforms **Surrogate** by 5% even in the one-shot evaluation protocol (see Table C.1), largely due to our incorporation of guidance gradients—entirely absent in all prior approaches—highlighting the unique strength of **DiffGrad**.

Note that the **surrogate** method shortens the trajectory and backpropagates through this shorter horizon, but retains the full graph over that path. Hence, one may argue that the attack success degradation is an acceptable tradeoff when using the **surrogate** method given the potential speedup. Yet, in *DBP*, memory—not FLOPs—is the bottleneck; Checkpointing (e.g., our **DiffGrad**) recomputes individual path segments, yielding exact gradients with lower peak memory. **Surrogate** may improve speed but not memory pressure, and its truncation can miss dependencies, weakening gradients as shown above. In practice, attackers precompute adversarial examples offline since *DBP*'s latency makes real-time adversarial generation infeasible either way, so runtime savings alone do not translate to a practical advantage.

All in all, the results demonstrate the fragility of *DBP* in the face of accurate gradients and highlight the issues in previous works' backpropagation. Nonetheless, this one-shot evaluation protocol remains problematic, as previously stated, leading to an inflated robustness estimate. In fact, under more realistic settings (see §5.4.2), we demonstrate that this gradient-based attack almost entirely defeats *DBP* when only one sample is used to predict the label (i.e., Wor.Rob).

C.5 Evaluations with Liu et al. [162]'s Fixed *AutoAttack*

Liu et al. [162], like **DiffHammer** [251] and our own analysis, note that evaluating a single purification at attack termination inflates robustness scores. Liu et al. [162] address this by evaluating 20 replicas of the final *AE*, declaring success if any is misclassified. While similar in spirit to the Wor.Rob metric we consider (see §5.3.3), this protocol is more limited: it evaluates only at the final step, whereas Wor.Rob evaluates N copies at each

attack iteration. Accordingly, Liu et al. [162] group their method with one-shot evaluations, providing a slightly more realistic assessment.

Models	Pur.	Gradient Method	Cl-Acc %	Rob-Acc %
WideResNet-28-10	*DiffPure* [182]	**Full** (Liu et al. [162])	89.26	56.25
		Full-DiffGrad	89.46	**30.86**
	GDMP [250]	**Full** (Liu et al. [162])	91.80	40.97
		Full-DiffGrad	93.36	**10.55**
WideResNet-70-16	*DiffPure* [182]	**Full-DiffGrad**	89.06	**35.16**
	GDMP [250]	**Full-DiffGrad**	91.8	**8.59**

Table C.2: *AA*-ℓ_∞ comparison on ***CIFAR-10*** **under Liu et al. [162]'s protocol (i.e., Fixed *AutoAttack*)** (ϵ_∞=8/255).

We replicate their setup using **DiffGrad** (Table C.2), running 20 iterations with $N = 10$ EOT samples per step and evaluating 20 final replicas. Under this protocol, our improvements are stark. On WideResNet-28-10, we reduce Rob-Acc by 25.39% and 30.42% for *DiffPure* and *GDMP*, bringing final accuracy to 30.86% and 10.55%, respectively. These results confirm the superiority of **DiffGrad**'s gradients and expose *DBP*'s realistic vulnerability. We observe similar results on WideResNet-70-16 (not evaluated in [162]).

C.6 Ablation Study

To assess the impact of the number of evaluation samples N, we test both single-purification and majority-vote (MV) settings with $N \in \{1, 10, 128\}$. Our goal is to highlight the brittleness of *DBP*'s standard deployment, which classifies based on a single purified copy. As explained in §5.3.3, in stateless setups (e.g., phishing, spam, CSAM), adversaries can resubmit identical queries indefinitely. Since randomized defenses like *DBP* can fail along certain stochastic paths, any non-negligible misclassification probability compounds with repeated attempts—see §5.3.3. Although *DBP* assumes this probability is negligible, our proof in §5.3.1, results in §5.4, and the ablation experiments here contradict this. Furthermore, we wish to showcase the advantages of our proposed majority-vote *(MV)* setting that strives to mitigate this vulnerability by, instead, classifying based on expectation of the randomized defense.

We evaluate *DiffPure* and *GDMP* on ***CIFAR-10*** using WideResNet-70-16 and (our) *AA*-ℓ_∞ (with $\epsilon = 8/255$ and 100 iterations). For $N = 1$ and $N = 10$, we use 10 EOT samples;

for $N = 128$, we reuse the same 128 samples for both EOT and evaluation. This improves gradient quality, thus strengthening attacks, yet still leads to higher *MV* robustness, proving the superiority of our proposed method despite the use of more accurate gradients.

For Wor.Rob, clean accuracy (Cl-Acc) remains constant across all N values since we always compute it using a single purified copy ($N = 1$), independent of the number N of samples used in the attack. This reflects the actual standard single-purification deployment of *DBP*, where the defender classifies a single output, while the attacker may retry multiple times. Hence, for Rob-Acc we consider a batch of multiple samples (the corresponding N in that row) and then declare attack success if a single misclassification occurs (after any given optimization step), revealing the gap between measured and effective robustness. In contrast, MV.Rob clean accuracy varies with N, as the prediction always aggregates over N purified samples, consistent with our proposed deployment.

Pur.	#Samples	Wor.Rob %		MV.Rob %	
		Cl-Acc	Rob-Acc	Cl-Acc	Rob-Acc
DiffPure [182]	**N=1**	89.06	35.16	89.06	35.16
	N=10		17.19	91.02	39.45
	N=128		17.58	92.19	47.72
GDMP [250]	**N=1**	91.8	8.59	91.8	8.59
	N=10		7.03	92.19	16.8
	N=128		5.47	92.19	32.81

Table C.3: *AA*-ℓ_∞ Performance under various evaluation sample counts

Table C.3 confirms that Wor.Rob consistently declines with N, revealing the illusion of robustness under single evaluation. For instance, *DiffPure* shows a drop from 35.16% (at $N = 1$) to 17.58% at $N = 128$ (far lower than the inflated numbers reported in previous studies—see Appendix C.4), confirming that many stochastic paths yield incorrect predictions, as our properly implemented gradient-based attack (see §5.3.2) lowers the expected classification confidence, making such failure modes more likely. For *GDMP*, we observe a similar trend.

In contrast, MV.Rob improves with N, rising from 35.16% to 47.72% for *DiffPure*, and from 8.59% to 32.81% for *GDMP*. This affirms *MV* as a more stable and accurate evaluation method that must be adopted as the de facto standard for *DBP* evaluations in the future.

Yet, despite *MV*'s benefits, our gradient-based *AA*-ℓ_∞ attack still degrades robustness, which never exceeds 50%, validating our theoretical finding from §5.3.1 that such attacks repurpose *DBP* into an adversarial distribution generator.

Computational Cost. While a larger N increases computational overhead, we identify $N = 128$ as the max batch size fitting in a single *A100 GPU* run, requiring $\sim 26.53s$ for inference, with $N = 10$ offering a practical middle ground ($6.54s$ vs. $5.29s$ for $N = 1$). Hence, for practicality and if throughput is critical, one should opt for $N = 10$. However, in high-stakes systems where latency is not crucial, a larger N is favorable. Yet, additional tests (not shown) suggest MV.Rob plateaus near $N = 128$. Note that these latencies refer to purification inference alone (classification excluded). During attacks (not standard deployment), there is also the cost of backpropagation. When accounting for the cost of classification and backpropagation, the latency becomes $\sim$ 17s for $N = 1$ and $N = 10$, compared to $\sim$ 53s for $N = 128$. As explained in §5.3.2, our design allows purifying multiple stochastic copies of the same sample in one step, in contrast to previous work. This reduces the runtime per EOT gradient from up to $N \times 17$s—where N stochastic purifications are run serially—to just T_Ns, where T_N is the latency incurred by purifying N samples in a batch, yielding a speedup of up to $N \times 17/T_N$ for single-sample attacks, which amounts to $41.06\times$ for $N = 128$ allowing for a considerable speedup when the objective is to utilize many EOT samples to obtain accurate gradient estimates. For batch attacks, prior methods purify one stochastic copy per sample per EOT step and require all samples to converge before proceeding to the next batch of inputs, blocking early termination. In contrast, **DiffGrad** purifies N copies of a single sample in parallel, allowing samples to terminate independently. This greatly improves overall throughput by freeing compute sooner, especially when convergence varies across inputs.

ImageNet** timings:* While we limit our ablation studies to ***CIFAR-10 for feasibility, we provide ***ImageNet*** timings for completeness. At ***ImageNet*** resolution on a 40 GB *A100 GPU*, *DBP* inference (forward propagation only) requires $\sim$ 7s for $N = 1$ and $\sim$ 25s for $N = 8$. A single attack iteration (i.e., forward and backward propagation) lasts $\sim$ 16s ($N = 1$) and $\sim$ 75s ($N = 8$). Thus, batching EOT copies yields a significant speedup over serial purification for both attacks and inference, but models and activations dominate cost at this scale, so batching returns diminish faster than on ***CIFAR-10***.

Recommendations for Choosing The Batch Size N. We found $N = 128$ for ***CIFAR-10*** and $N = 8$ for ***ImageNet*** to be the largest feasible batch sizes to fit into the 40 GB *A100 GPU* memory, making larger batches impractical for large-scale evaluations currently. That said, control experiments with larger N values for both datasets (split into batches) showed negligible robustness or attack performance gains, indicating saturation.

Following the above discussion, we recommend $N = 10$ for ***CIFAR-10*** as a practical default and $N = 64 - 128$ for high-stakes inference (MV) and highly accurate gradients

when evaluating new attack methods such as our *LF* strategy—see §5.5. For ***ImageNet***, we recommend $N = 8$ for similar reasons; larger N values remain preferable when added latency is acceptable.

C.7 Details on Our Low-Frequency (*LF*) Adversarial Optimization Strategy

C.7.1 Attack Hyperparameters

Through experimentation, we found that the loss balancing constants $c = 10^8$ for ***CIFAR-10*** and $c = 10^4$ for ***ImageNet*** lead to the fastest convergence rates and selected these values accordingly (although other choices are also possible). The filter network architecture for ***ImageNet*** is identical to that used in *UnMarker* (see Appendix B.1). For ***CIFAR-10***, since the images are much smaller, we found the original architecture unnecessarily costly and often prevents convergence since larger filters group pixels from distant regions together in this case, easily violating visual constraints upon each update, resulting in the *LPIPS* condition being violated. Thus, we opt for a more compact network that includes filters with smaller dimensions, which was chosen based on similar considerations to *UnMarker*'s [130] (see §4.5.2), allowing us to explore several interpolation options. The chosen architecture for ***CIFAR-10*** includes 4 filters, whose dimensions are: $(5, 5)$, $(7, 7)$, $(5, 5)$, $(3, 3)$. We use fixed learning rates of 0.008 for the direct modifier $\boldsymbol{\delta}$ and 0.05 for the filters' weights, optimized using Adam. The remaining hyperparameters were left unchanged compared to Appendix B.1.

C.7.2 Pseudo-Code

The pseudo-code for our low-frequency (*LF*) strategy (see §5.5) is in Algorithm C.3. Importantly, as each $\boldsymbol{\kappa}_b^{i,j}$'s values should be non-negative and sum to 1, the values for each such per-pixel kernel are effectively obtained by ***softmax***ing the learned weights. Initially, the modifier $\hat{\boldsymbol{\delta}}$ is initialized to $\mathbf{0}$ and the weights $\{\widehat{\boldsymbol{\theta}}_{\boldsymbol{\kappa}_b^*}\}$ are selected s.t. the filters perform the identity function (line 1). As a result, the attack starts with $\boldsymbol{x}_{adv} = \boldsymbol{x}$ that is iteratively optimized. Similar to *C & W* [32], we directly optimize $\boldsymbol{x}_{adv}$ (i.e., using the modifier $\hat{\boldsymbol{\delta}}$) in the ***arctanh*** space, meaning we transform the sample first to this space by applying ***arctanh*** (after scaling $\boldsymbol{x}_{adv}$ to ***arctanh***'s valid range $[-1, 1]$) where $\hat{\boldsymbol{\delta}}$ is added and then restore the outcome to the original problem space (i.e., $[min_val, max_val]$,

Algorithm C.3 Low-Frequency (LF) Adversarial Optimization

Require: Sample $\boldsymbol{x}$, Model (classifier) $\mathcal{M}$, DBP pipeline D, Loss function ℓ, True label y of $\boldsymbol{x}$, Perceptual loss $\boldsymbol{LPIPS}$, $\boldsymbol{LPIPS}$ threshold τ_p, Filter architecture $\prod_{\boldsymbol{OF}}^{B}$, Balancing constant c, Iterations max_iters, Success condition $\boldsymbol{Cond}$, Filter weights learning rate $lr_{\boldsymbol{OF}}$, Modifier learning rate $lr_{\boldsymbol{\delta}}$, Number of purified copies n, Number of $\boldsymbol{EoT}$ eot_steps, Input range limits (min_val, max_val)

$\{\widehat{\boldsymbol{\theta}}_{\mathcal{K}_b^*}\} \leftarrow \boldsymbol{identity_weights}(\prod_{\boldsymbol{OF}}^{B})$, $\hat{\boldsymbol{\delta}} \leftarrow \mathbf{0}$ /* Initialize attack parameters. */

$\boldsymbol{Optim} \leftarrow \text{Adam}([\{\widehat{\boldsymbol{\theta}}_{\mathcal{K}_b^*}\}, \hat{\boldsymbol{\delta}}], [lr_{\boldsymbol{OF}}, lr_{\boldsymbol{\delta}}])$

$\boldsymbol{x}_{inv} \leftarrow \boldsymbol{inv_scale_and_arctanh}(\ \backslash$
$\boldsymbol{x}, min_val, max_val)$ /* Scale $\boldsymbol{x}$ to $[-1, 1]$ and take $\boldsymbol{arctanh}$. */

for $i \leftarrow 1$ **to** $max_iters \cdot eot_steps$ **do**

  $\boldsymbol{x}_{adv} \leftarrow \prod_{\boldsymbol{OF}}^{B}(\boldsymbol{tanh_and_scale}(\ \backslash$
  $\boldsymbol{x}_{inv} + \hat{\boldsymbol{\delta}}, min_val, max_val)$ /* Generate adversarial input using new $\{\widehat{\boldsymbol{\theta}}_{\mathcal{K}_b^*}\}$ and $\hat{\boldsymbol{\delta}}$ via (*eq.* (5.3)) scaled to $[min_val, max_val]$. */

  $dist \leftarrow \boldsymbol{LPIPS}(\boldsymbol{x}, \boldsymbol{x}_{adv})$ /* Calculate perceptual distance. */

  $\hat{\boldsymbol{x}}_{adv}^0 \leftarrow D(\boldsymbol{repeat}(\boldsymbol{x}_{adv}, n))$ /* Get purified outputs. */

  $\boldsymbol{logits} \leftarrow \mathcal{M}(\hat{\boldsymbol{x}}_{adv}^0)$ /* Compute model output. */

  if $\boldsymbol{Cond}(\boldsymbol{logits}, y)$ **and** $dist \leqslant \tau_p$ **then**

    return $\boldsymbol{x_{adv}}$ /* Success. Return $\boldsymbol{x}_{adv}$. */

  end if

  $Objective \leftarrow \ell(\boldsymbol{logits}, y) + c \cdot \max(dist - \tau_p, 0)$ /* Compute loss (*eq.* (5.3)). */

  $Objective.\boldsymbol{backward}()$ /* Get gradients for parameters. */

  if $i \bmod eot_steps = 0$ **then**

    $\boldsymbol{Optim.step}()$ /* Update parameters. */

    $\boldsymbol{Optim.zero_grad}()$ /* Reset gradients. */

  end if

end for

return $\boldsymbol{x}$ /* Failure. Return original $\boldsymbol{x}$. */

which is typically $[0,\ 1]$) via the $\boldsymbol{tanh}$ operation. Further details on this method and its benefits can be found in [32]. All other steps correspond to the description brought in §5.5. Unlike §5.3.1, we assume the classifier $\mathcal{M}$ outputs the logit vector rather than the probabilities (i.e., we omit the $\boldsymbol{softmax}$ layer over its output, which may or may not be re-introduced by the loss ℓ), as is traditionally done for a variety of adversarial optimization strategies (e.g., *C & W* [32]) to avoid gradient vanishing. We use the known $\boldsymbol{max\text{-}margin}$ loss [32]— $\ell(\boldsymbol{logits}, y) = \boldsymbol{logits}[, : y] - \max_{j \neq y} \{\boldsymbol{logits}[, : j]\}$.

To average the gradients over multiple (N) paths as per the adaptive attack's requirements from §5.3.1, we generate several purified copies by repeating the sample $\boldsymbol{x}_{adv}$ under optimization n times before feeding it into the *DBP* pipeline (line 9). Here, n corresponds to N or the maximum number of copies we can fit into the *GPU*'s memory during a single run. However, as this n may be smaller than the desired N from §5.3.1 (i.e., number of *EoT* samples) that allows us to sufficiently eliminate the error in the computed gradient, we use gradient accumulation by only making updates to the optimizable parameters (and then resetting their gradients) every *eot_steps* iterations (lines 14-17). By doing so, the effective number of used copies becomes $n * eot_steps$, which can represent any N of choice that is divisible by n. Note that if n is not a divisor of N, we can always increase N until this condition is met, as a larger N can only enhance gradient accuracy). This also explains why the algorithm runs for $max_iters * eot_steps$ (line 4).

Finally, the condition $\boldsymbol{Cond}$ captures the threat model (either *SP* or *MV*): When the logits for the n copies are available together with the target label y, $\boldsymbol{Cond}$ outputs a success decision based on whether we seek misclassification for the majority of these purified copies or a single copy only. Note that, as explained in §5.4.2, we take the majority vote over the maximum number of copies we can fit into the *GPU* (i.e., n) for *MV*. As this choice was only made for practical considerations, one may desire to experiment with different configurations wherein another number of copies is used. Yet, this is easily achievable by simply modifying $\boldsymbol{Cond}$ accordingly: For instance, we may augment it with a history that saves the output logits over all *eot_steps* (during which $\boldsymbol{x}_{adv}$ is not updated). Then, the majority vote can be taken over all copies in this window. By increasing the number of *eot_steps*, we can use as many copies for the majority vote decision as desired in this case.

In addition to the precise gradient module **DiffGrad**, our *DiffBreak* toolkit provides the implementation of our *LF* strategy, as well as various other common methods (e.g., *AA* and *StAdv*), to enable robust and reliable evaluations of *DBP*. All strategies are optimized for performance to speed up computations via various techniques such as just-in-time (*JIT*) compilation. Our code is available at https://github.com/andrekassis/DiffBreak.

C.8 Example Attack Images

In Appendix C.8.1-C.8.10, we provide a variety of successful attack images that cause misclassification in the rigorous MV setting, generated using our low-frequency (LF) strategy against all systems considered in §5.5.1. For configurations that were also evaluated against AA-ℓ_∞ under the **same** MV setting (i.e., using the same sample counts N as used for LF in §5.5.1), we include successful AEs generated with this method for direct comparison. Specifically, for ***ImageNet***, both the LF attack in §5.5.1 and the AA attack in §5.4.2 were evaluated under MV with $N = 8$; hence, we include AA samples for ***ImageNet*** directly from §5.4.2. For ***CIFAR-10***, the AA experiments in §5.4.2 use the relaxed $N = 10$ setting. Therefore, we instead draw samples from the corresponding $N = 128$ AA experiments reported in Appendix C.6 to ensure a fair comparison under equal majority-vote conditions. Note that all samples are crafted using the parameters listed in §5.4.2 and §5.5.1. That is, $\tau_p = 0.05$ for LF and $\epsilon_\infty = 8/255$ for AA against ***CIFAR-10*** and $\epsilon_\infty = 4/255$ against ***ImageNet***.

For the configurations that were evaluated against both strategies, we provide two sets of samples: 1) Three triplets containing the original image, the AE generated using AA, and the AE crafted using LF. Importantly, all original samples in this set are inputs for which both methods can generate successful AEs, and we provide these to allow for a direct comparison between the two strategies' output quality on a sample-by-sample basis. Yet, as AA is inferior to our approach (LF), resulting in the systems retaining robustness on many inputs for which it fails to generate successful AEs under MV, it is essential to inspect LF's outputs on such more challenging samples to demonstrate that it still preserves quality despite its ability to fool the target classifiers. Thus, we include a second set of: 2) Three successful AEs generated with LF from inputs on which AA fails under MV. For the remaining configurations that were not evaluated against AA under the same MV sample counts from §5.5.1, we provide six successful AEs generated with LF.

In Appendix C.8.11, we present attack images generated by the non-norm-bounded *StAdv* [271] strategy under MV (with the above sample counts N). This method has demonstrated superior performance to norm-based techniques in the past against *DBP* [182] even in the absence of the correct exact gradients, indicating it could be a viable attack strategy with our gradient computation fixes, thereby making our LF approach unnecessary. Yet, previous evaluations only considered *StAdv* against *DBP* for ***CIFAR-10*** [182]. While we find *StAdv* capable of defeating all systems (for both ***CIFAR-10*** and ***ImageNet***), it leads to severe quality degradation when used to attack *DBP*-protected classifiers for high-resolution inputs (i.e., ***ImageNet***), leaving the AEs of no utility. Thus, we deem it unsuitable, excluding it from the main body of Chapter 5 accordingly. Further details are

in Appendix C.8.11.

All samples below are originally (without adversarial perturbations) correctly classified.

C.8.1 Attack Samples Generated Against *CIFAR-10*'s WideResNet-70-16 with *GDMP* Purification



Figure C.2: Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*.



Figure C.3: Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*.

C.8.2 Attack Samples Generated Against *CIFAR-10*'s WideResNet-28-10 with *GDMP* Purification

Note: While WideResNet-28-10 was not evaluated against AA under MV with $N = 128$ *in Appendix C.6, we include AA samples here to complement our CIFAR-10 analyses—our primary benchmark for AA-based evaluation. Among the two main purification paradigms considered (GDMP and DiffPure), we randomly selected GDMP for this illustrative example.*

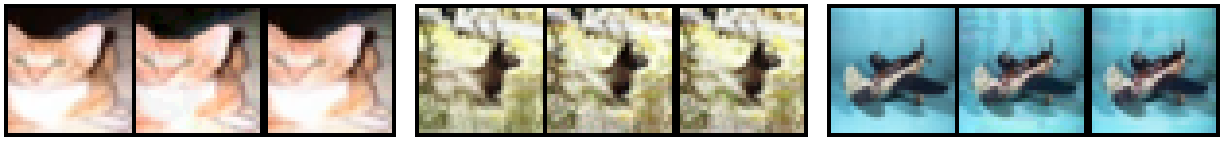


Figure C.4: Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*.

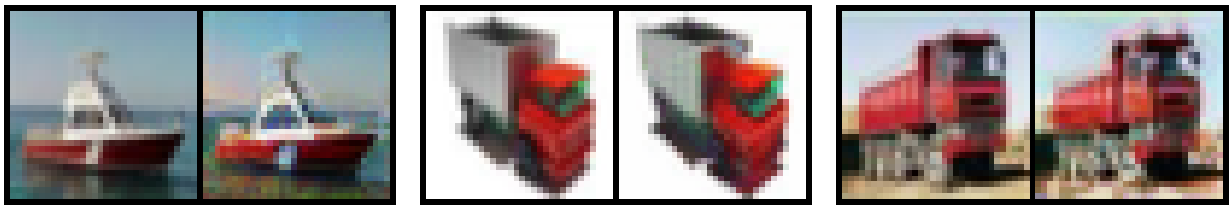


Figure C.5: Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*.

C.8.3 Attack Samples Generated Against *CIFAR-10*'s WideResNet-70-16 with *DiffPure* Purification



Figure C.6: Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*.

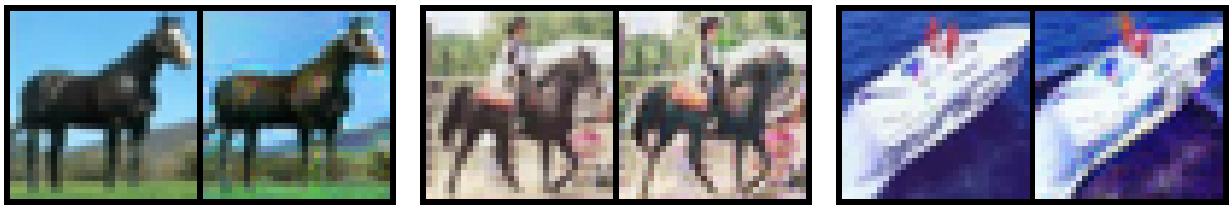


Figure C.7: Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*.

C.8.4 Attack Samples Generated Against *CIFAR-10*'s WideResNet-28-10 with *DiffPure* Purification

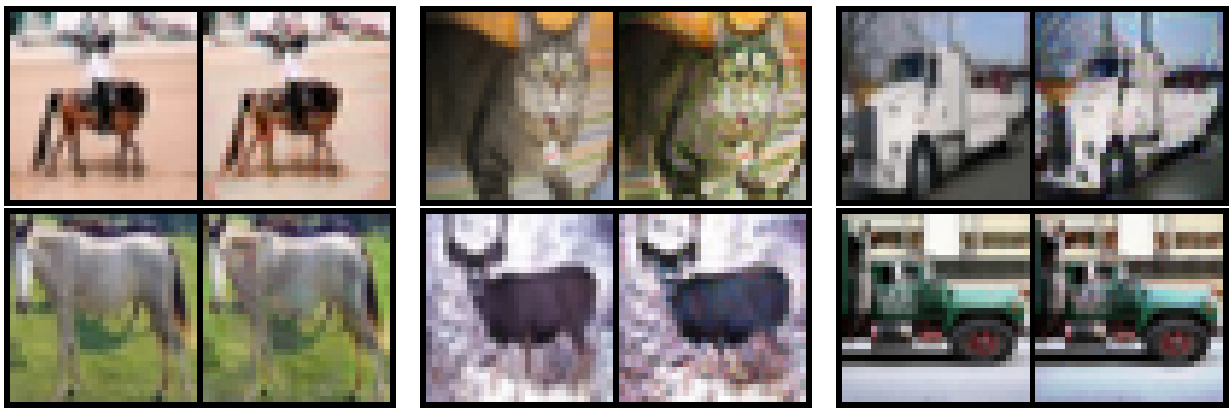


Figure C.8: Successful attacks generated with *LF*. Left -original image. Right - *LF*.

C.8.5 Attack Samples Generated Against *ImageNet*'s DeiT-S with *GDMP* Purification

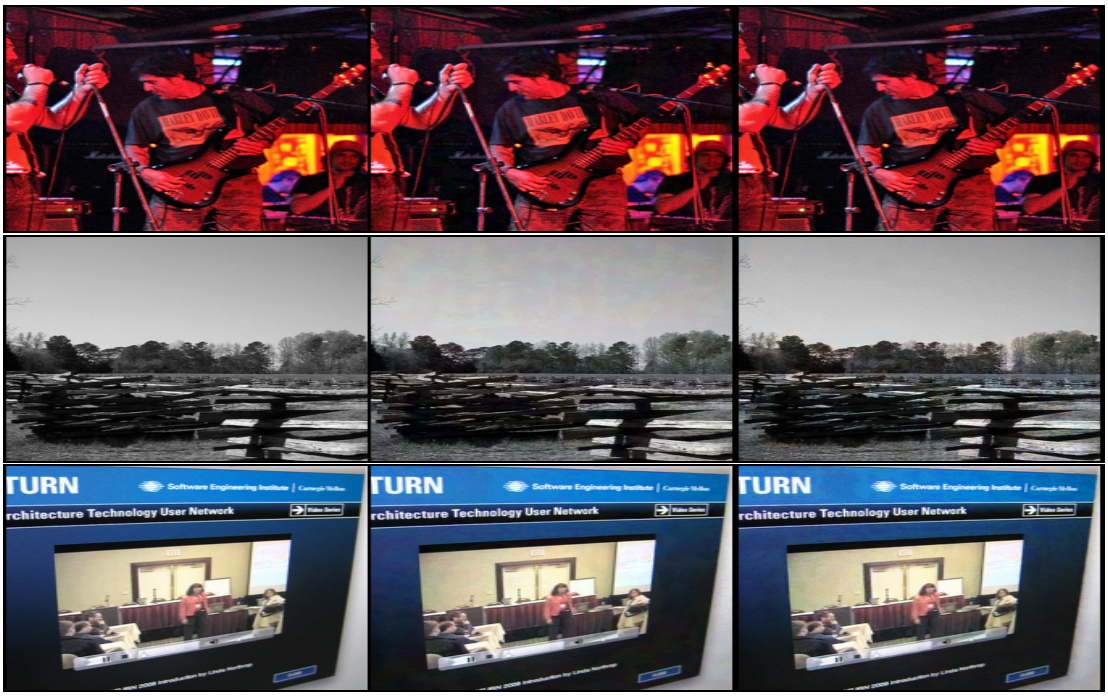


Figure C.9: Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*.



Figure C.10: Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*.

C.8.6 Attack Samples Generated Against *ImageNet*'s WideResNet-50-2 with *GDMP* Purification



Figure C.11: Successful attacks generated with *LF*. Left -original image. Right - *LF*.

C.8.7 Attack Samples Generated Against *ImageNet*'s ResNet-50 with *GDMP* Purification

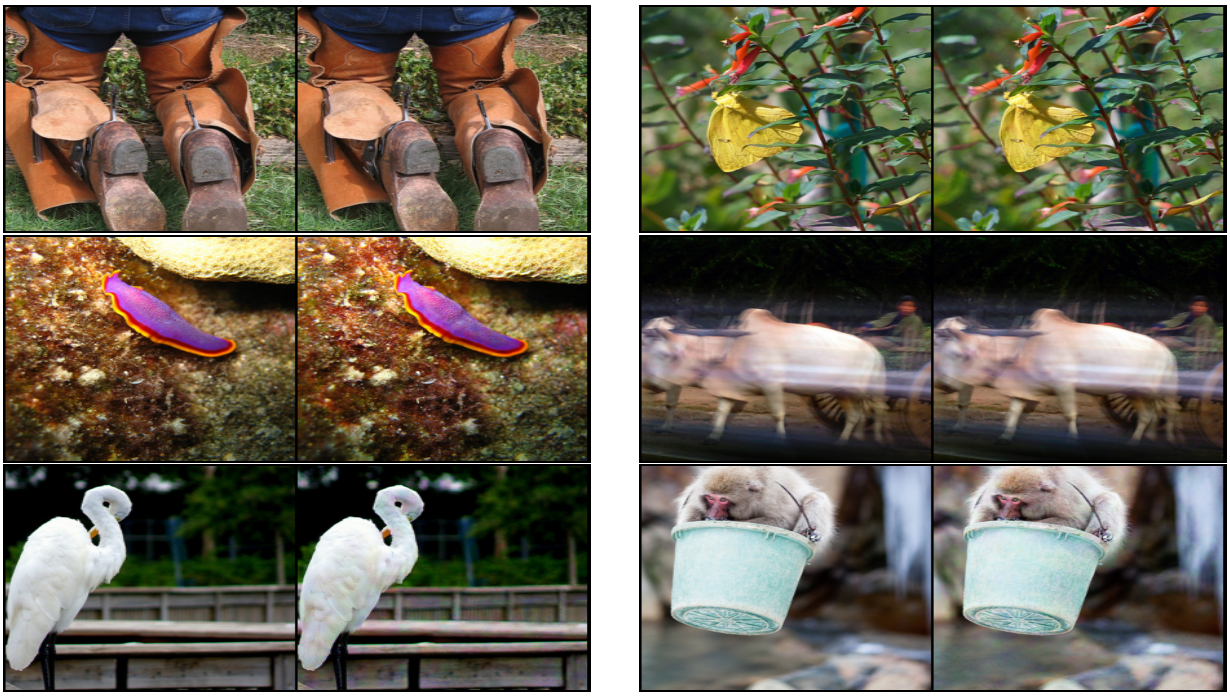


Figure C.12: Successful attacks generated with *LF*. Left -original image. Right - *LF*.

C.8.8 Attack Samples Generated Against *ImageNet*'s DeiT-S with *DiffPure* Purification



Figure C.13: Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*.

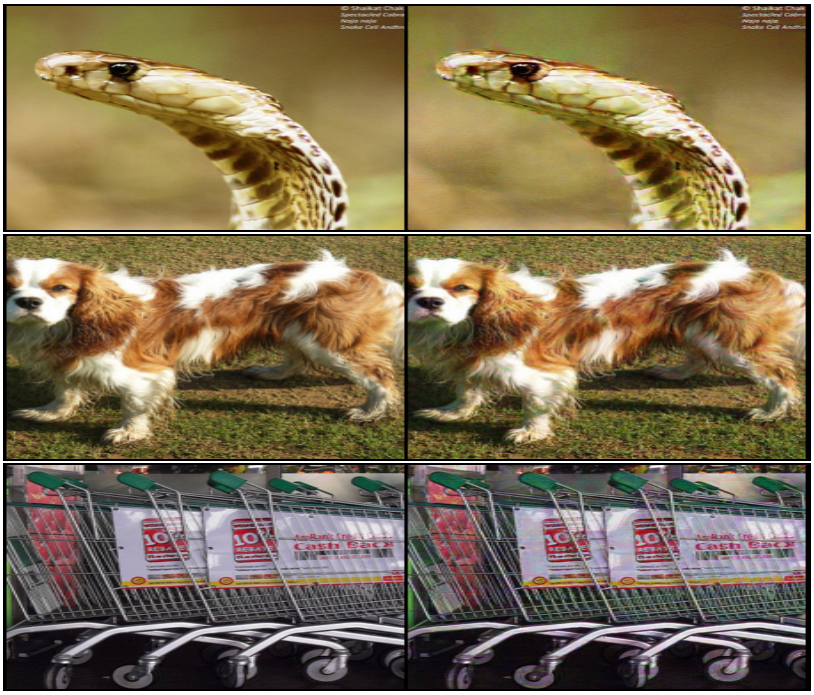


Figure C.14: Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*.

C.8.9 Attack Samples Generated Against *ImageNet*'s WideResNet-50-2 with *DiffPure* Purification

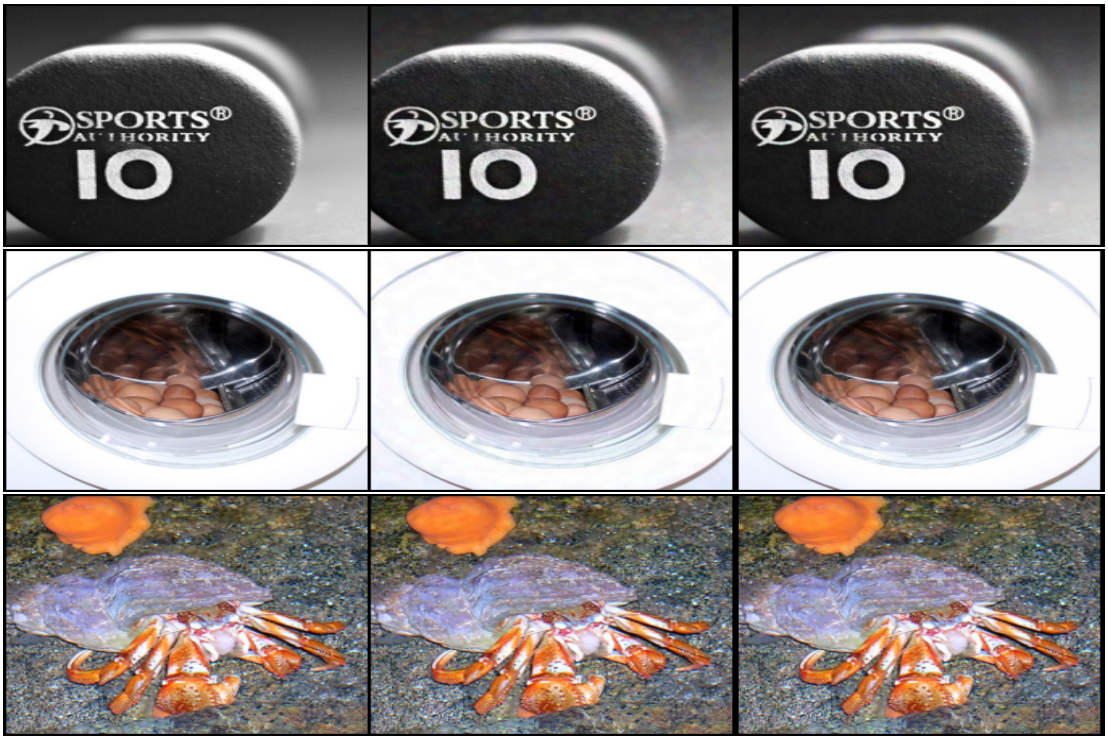


Figure C.15: Successful attacks generated by *LF* and *AA*-ℓ_∞. Left -original image. Middle - *AA*. Right - *LF*.

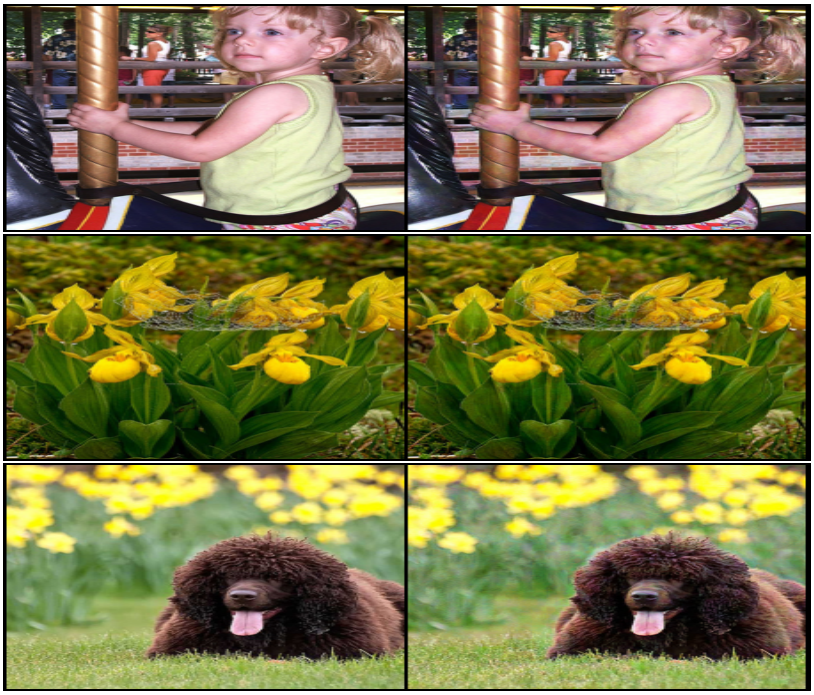


Figure C.16: Successful *LF* attacks on inputs for which *AA*-ℓ_∞ fails. Left - original image. Right - *LF*.

C.8.10 Attack Samples Generated Against *ImageNet*'s ResNet-50 with *DiffPure* Purification

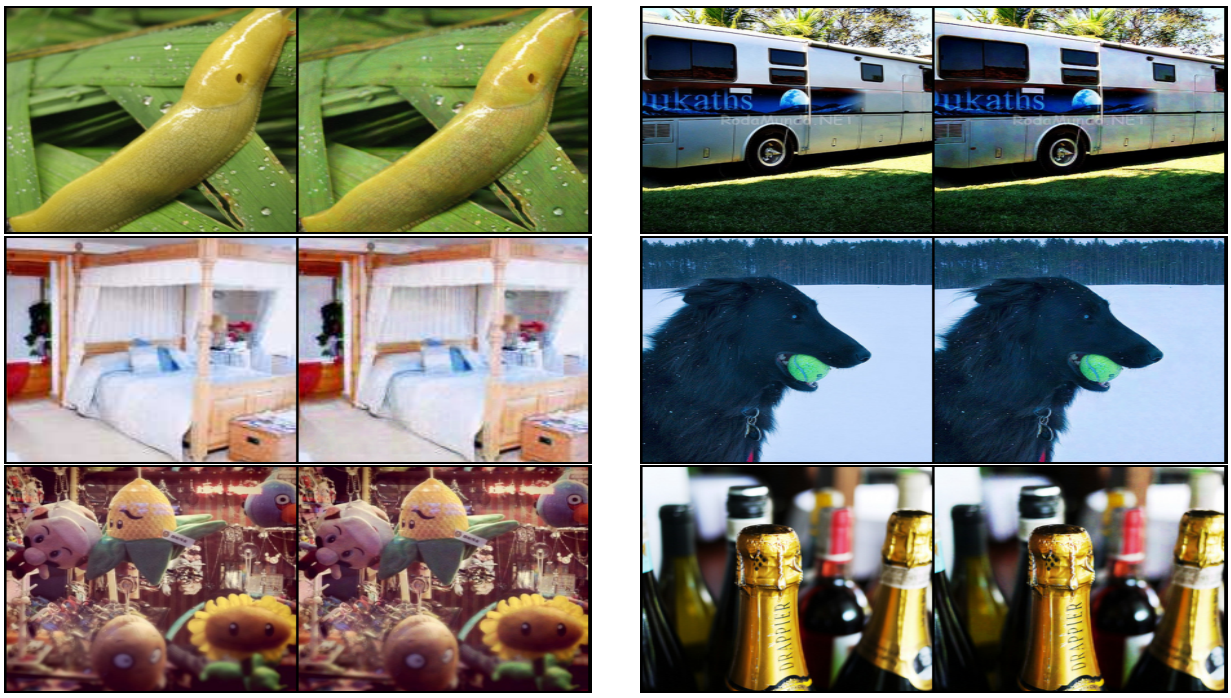


Figure C.17: Successful attacks generated with *LF*. Left -original image. Right - *LF*.

C.8.11 Quality Comparison with *StAdv*

We found *StAdv* capable of generating outputs that defeat *DBP* even under *MV*. However, it is not suitable for targeting *DBP*-defended classifiers that operate on high-resolution images. The reason is that *StAdv* performs spatial transformations that relocate the different pixels. Thus, its changes quickly become visible when applied excessively. Due to the considerable stochasticity of *DBP* (see §5.4.2), the required displacements (especially in the *MV* setting) are significant, which in turn can severely impact quality. For low-resolution inputs (e.g., ***CIFAR-10***), *StAdv* can still be effective, with the quality degradation remaining unnoticeable due to the size of the images that renders them blurry by default, masking *StAdv*'s effects. For high-resolution inputs, the degradation is substantial, leaving the outputs useless as stealthiness is a key requirement from practical *AE*s [130]. *StAdv*'s

successfully misclassified samples (under MV) below prove these claims. We use **Full-DiffGrad** for backpropagation and run *StAdv* with its default parameters [271]. When the parameters are changed to better retain quality, *StAdv* ceases to converge for ***ImageNet***, making it of no use. All provided samples are originally correctly classified.



Figure C.18: *StAdv* attacks against ***CIFAR-10***'s WideResNet-70-16 with *GDMP* purification. Left - original image. Right - *StAdv*.

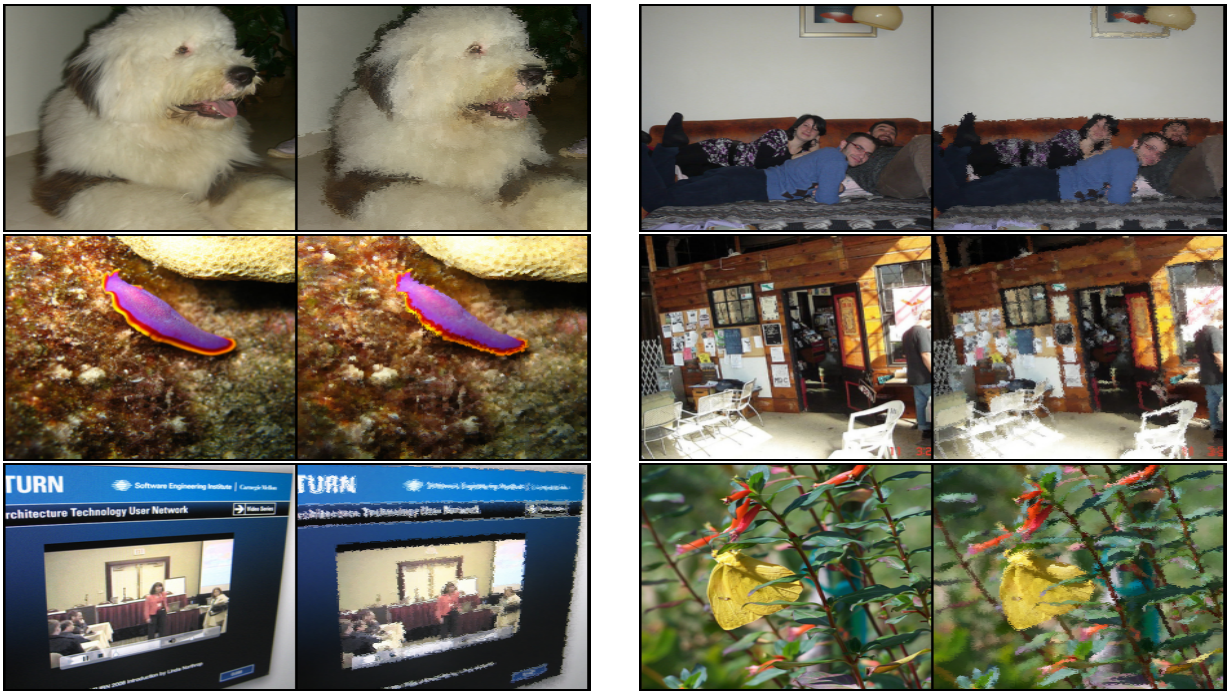


Figure C.19: *StAdv* attacks against ***ImageNet***'s DeiT-S with *DiffPure* purification. Left - original image. Right - *StAdv*.

Appendix D

KEYSHIFT: Key-Induced Boundary Shifts for Adversarial Robustness

D.1 Design & Training

Below we provide details on our preprocessor architecture and its training procedure. We first present our ConvolveCond conditioning mechanism and explain how we apply simple modifications to standard U-Nets to integrate this operator. Finally, we describe the curriculum training algorithm.

D.1.1 ConvolveCond

Our conditioning logic ConvolveCond (Algorithm D.1) is modular and can be introduced into any residual block of an existing U-Net. Each conditioning point applies ConvolveCond to the current feature $\boldsymbol{x}$ and key state $\boldsymbol{z}$ tensors, producing an updated pair $(\boldsymbol{x}', \boldsymbol{z}')$. $\boldsymbol{z}$ is first projected (line 1) to match the feature channels and resized (line 2) to the feature resolution (since U-Nets perform up/down sampling and channel changes); the resulting tensors are concatenated (line 3) and processed by a small convolutional network f (line 4)—we instantiate f as $\texttt{Conv}_{3\times3} \to \texttt{SiLU} \to \texttt{Conv}_{3\times3} \to \texttt{SiLU}$, mapping its input u as $2C \to \mathit{hidden} \to 2C$, where $\mathit{hidden} = 128$ (but other alternatives for f could be explored). The output is split into new feature and key tensors (line 5), with the latter compressed back to a single channel (line 6) and resized to the original key dimensions (line 7). Then, both are propagated to subsequent layers (line 8).

Algorithm D.1 ConvolveCond: Coupled feature–key update

Require: Feature map $\boldsymbol{x} \in \mathbb{R}^{B\times C\times H_x\times W_x}$, key state $\boldsymbol{z} \in \mathbb{R}^{B\times 1\times H\times W}$
Ensure: Updated $(\boldsymbol{x}', \boldsymbol{z}')$

1: $\boldsymbol{z}_1 \leftarrow \mathrm{Conv}_{1\times 1}^{(1\to C)}(\boldsymbol{z})$
2: $\boldsymbol{z}_1 \leftarrow \mathrm{Resize}(\boldsymbol{z}_1, H_x, W_x)$ ▷ up/downsample to match $\boldsymbol{x}$
3: $\boldsymbol{u} \leftarrow \mathrm{Concat}(\boldsymbol{x}, \boldsymbol{z}_1)$ ▷ concat along channel dimension
4: $\boldsymbol{u} \leftarrow f(\boldsymbol{u})$ ▷ small conv-net
5: $\boldsymbol{x}',\ \tilde{\boldsymbol{z}} \leftarrow \boldsymbol{u}[:, 1{:}C, :, :],\ \boldsymbol{u}[:, C{+}1{:}, :, :]$
6: $\boldsymbol{z}' \leftarrow k(\tilde{\boldsymbol{z}})$ ▷ $C \to 1$ convolution (project to 1 channel)
7: $\boldsymbol{z}' \leftarrow \mathrm{Resize}(\boldsymbol{z}', H, W)$
8: **return** $\boldsymbol{x}',\ \boldsymbol{z}'$

Entanglement. The key state is repeatedly mixed with the feature (input) representation and updated across layers by ConvolveCond (see below). Consequently, private behavior depends on the evolution of a joint, entangled feature–key state that is always processed by the same computational conditioning units. Hence, techniques attempting to manipulate the key state's processing units will simultaneously affect the input computation pathway, likely destabilizing predictions rather than revealing private behavior, resisting architectural analysis attacks as explained in §6.3.2.

D.1.2 U-Net Integration

The preprocessor P is implemented using a U-Net backbone. U-Nets consist of a hierarchy of feature transformations combining convolutions, normalization, residual connections, and resolution changes. As explained in §6.3.2, the only architectural element we modify is the residual block. Each residual block is replaced with a *conditional residual block* that additionally receives and propagates the key state through the network. Algorithm D.2 summarizes the resulting block. Aside from the addition of the key state and two ConvolveCond operations, it follows the standard residual formulation. This modification is backbone-agnostic: any U-Net architecture using residual blocks can be conditioned by replacing each block with the conditional variant described below and propagating the key state throughout the network.

The operations in black correspond to the standard residual block structure. The *Skip* operation (line 1) denotes the residual path, which is typically the identity or a 1×1 convolution used to match channel dimensions between the input and output of the block. The result of this operation is stored and later combined with the transformed features.

Algorithm D.2 Conditional residual block

Require: Feature tensor $\boldsymbol{x}$, key state $\boldsymbol{z}$
Ensure: Updated feature tensor $\boldsymbol{x}'$, updated key state $\boldsymbol{z}'$

1: $\boldsymbol{r} \leftarrow \text{Skip}(\boldsymbol{x})$
2: $\boldsymbol{h} \leftarrow \text{PreOps}(\boldsymbol{x})$ ▷ norm, activation, dropout, etc.
3: $\boldsymbol{h} \leftarrow \text{Conv}_1(\boldsymbol{h})$
4: $\boldsymbol{h}, \boldsymbol{z}' \leftarrow \textsc{ConvolveCond}_1(\boldsymbol{h}, \boldsymbol{z})$
5: $\boldsymbol{h} \leftarrow \text{BlockOps}(\boldsymbol{h})$ ▷ norm, activation, dropout, convolutions, etc.
6: $\boldsymbol{h} \leftarrow \text{Conv}_2(\boldsymbol{h})$
7: $\boldsymbol{h}, \boldsymbol{z}' \leftarrow \textsc{ConvolveCond}_2(\boldsymbol{h}, \boldsymbol{z}')$
8: $\boldsymbol{h} \leftarrow \text{PostOps}(\boldsymbol{h})$ ▷ norm, activation, dropout, etc.
9: $\boldsymbol{x}' \leftarrow \text{Scale}(\boldsymbol{r} + \boldsymbol{h})$
10: **return** $\boldsymbol{x}', \boldsymbol{z}'$

PreOps (line 2), *BlockOps* (line 5), and *PostOps* (line 8) abstract the sequence of standard operations in residual blocks, including normalization layers, activations, dropout, and additional convolutions. These may vary slightly across implementations but do not affect the structure itself. The *Scale* operation (line 9) optionally rescales the residual merge, which is commonly used in U-Nets for numerical stability.

We extend the block to operate on a pair $(\boldsymbol{x}, \boldsymbol{z})$ rather than only the feature tensor $\boldsymbol{x}$. The gray lines indicate the two conditioning points where the ConvolveCond operator (Algorithm D.1) is applied. The first occurs immediately after the initial convolution (line 4), allowing the key to influence the early feature transformation. The second occurs after the final convolution (line 7) before the residual merge. Each invocation jointly updates the feature representation and the propagated key state, producing the pair $(\boldsymbol{x}', \boldsymbol{z}')$. Between these conditioning points, operations of the residual block remain unchanged from the standard architecture. Each of the two ConvolveCond operators has its unique parameters. This design entangles the two signals (see Appendix D.1.1): the output features depend on both the current feature representation and the key state, while the key state itself is updated based on the resulting activations as well.

D.1.3 Preprocessor Training

This subsection completes the description of the preprocessor training procedure. We first detail the adversarial objective, and then present the full training algorithm that optimizes all objectives introduced in §6.3.2.

Adversarial Objective

While we rely on label permutation to separate public and private behaviors, both utilize a shared architecture (parameters), allowing some *AE*s to transfer across modes. We reduce such shared directions via:

$$\mathcal{L}_{\text{adv}} = \ell(\mathcal{M}(P(\boldsymbol{x}_{adv}, \boldsymbol{z}^{\star})), \pi(y)) \,,$$

$\boldsymbol{x}_{adv}$ is an *AE* crafted directly against $\mathcal{M}$ via *PGD* [141] (ℓ_∞). Since public mode collapses to identity, *AE*s can be precomputed against the fixed classifier $\mathcal{M}$, avoiding the significant online cost of *AT*. We found *PGD* suffices to reduce adversarial overlap between modes and strengthen robustness to diverse attacks. We apply $\mathcal{L}_{\text{adv}}$ only when public-mode *AE*s measurably degrade private-mode robustness—see Appendix D.2.

Curriculum Training

Algorithm D.3 summarizes the preprocessor's training procedure. The classifier $\mathcal{M}$ remains frozen so that all private behavior is mediated solely through the preprocessor. Training proceeds in a staged curriculum that incrementally introduces the objectives from §6.3.2 based on whether all previous objectives have converged at the end of any training iteration (epoch).

Stage 1 (Private). Each iteration first enforces the *private-mode objective* (lines 6–8). The secret key $\boldsymbol{z}^{\star}$ is repeated B times across the batch and $P(\boldsymbol{x}, \boldsymbol{z}^{\star})$ is required to produce permuted labels $\pi(y)$ through the fixed classifier $\mathcal{M}$. Once this objective converges, training advances to the *public-collapse stage*.

Stage 2 (Public). In this phase (lines 9–12), keys are sampled (Monte Carlo) from the high-entropy distribution $\mathcal{Z}$ and the preprocessor is trained to collapse to the identity mapping, enforcing $P(\boldsymbol{x}, \boldsymbol{z}) \approx \boldsymbol{x}$ for non-secret keys.

Stage 3 (Adv). After both behaviors are established, the optional adversarial objective is introduced (lines 14–17), which trains P on *AE*s generated using *PGD* [141] against the

Algorithm D.3 Curriculum training of keyed preprocessor P

Require: Dataset $\mathcal{D}$, secret key $\boldsymbol{z}^\star$, permutation π, classifier $\mathcal{M}$, preprocessor P, training hyperparameters

Freeze $\mathcal{M}$; initialize P
stage $\leftarrow$ PRIVATE
for epoch $= 1$ to T **do**
  for batch $(\boldsymbol{x}, y)$ in $\mathcal{D}$ **do**
    $\mathcal{L} \leftarrow 0$
    # Private-mode objective
    $\hat{\boldsymbol{x}}_{priv} \leftarrow P(\boldsymbol{x}, \mathrm{Repeat}(\boldsymbol{z}^\star, B))$
    $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{\mathrm{priv}}\, \ell(\mathcal{M}(\hat{\boldsymbol{x}}_{priv}), \pi(y))$
    if stage $\in \{$PUBLIC, ADV, ROBUST$\}$ **then**
    # Public-collapse objective
      sample $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_B \sim \mathcal{Z}$
      $\hat{\boldsymbol{x}}_{pub} \leftarrow P(\boldsymbol{x}, \mathrm{Concat}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_B))$
      $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{\mathrm{pub}} \|\hat{\boldsymbol{x}}_{pub} - \boldsymbol{x}\|^2$
    end if
    if stage $\in \{$ADV, ROBUST$\}$ and *adv_obj* **then**
    # Optional adversarial regularization objective
      $\boldsymbol{x}_{adv} \leftarrow PGD(\boldsymbol{x}, \mathcal{M})$
      $\hat{\boldsymbol{x}}_{adv} \leftarrow P(\boldsymbol{x}_{adv}, \mathrm{Repeat}(\boldsymbol{z}^\star, B))$
      $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{\mathrm{adv}}\, \ell(\mathcal{M}(\hat{\boldsymbol{x}}_{adv}), \pi(y))$
    end if
    if stage $=$ ROBUST **then**
    # Robustness objective
      $\boldsymbol{z}' \leftarrow \mathrm{OptimizeKeyCandidate}(\mathcal{D}, S_{\mathrm{rob}})$ /* Inner key-recovery search */
      $\hat{\boldsymbol{x}}_{\mathrm{rob}} \leftarrow P(\boldsymbol{x}, \mathrm{Repeat}(\boldsymbol{z}', B))$
      $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{\mathrm{rob}} \big(\|\hat{\boldsymbol{x}}_{rob} - \boldsymbol{x}\|^2 + c_{\mathrm{rob}} \ell(\mathcal{M}(\hat{\boldsymbol{x}}_{rob}), y) \big)$
      if *early_termination* and CurrentStageSatisfied() **then**
        return P
      end if
    end if
    update P using $\nabla \mathcal{L}$
  end for
  if CurrentStageSatisfied() **then**
    advance *stage*
  end if
end for
return P

frozen classifier. This stage reduces perturbation directions that transfer between public

and private modes (see Appendix D.1.3).

Algorithm D.3 abstracts the decision process regarding whether this stage should be initiated through adv_obj. In practice, this variable is determined through the following procedure, omitted from the algorithm for brevity. After the private and public objectives have converged, we generate AEs in public mode using PGD (once) on a subset of the training data (1024 samples) and evaluate their transferability to private mode. If these perturbations noticeably degrade private-mode accuracy (i.e., accuracy drops below a preset threshold—see Appendix D.2), we enable the adversarial objective and include this stage. Otherwise, training proceeds directly to the next stage. Moreover, Algorithm D.3 presents the adversarial objective as if AEs were generated during training. In practice, this is unnecessary. As discussed in Appendix D.1.3, public mode collapses to the identity, allowing AEs to be generated directly against the classifier $\mathcal{M}$. Because $\mathcal{M}$ is fixed throughout training, these AEs can be precomputed before optimization begins. This avoids the substantial computational overhead that would arise from generating AEs online at each training step.

Stage 4 (Robust). Finally, robustness training against key-recovery attacks is performed (lines 19–21). At each training step (batch), an inner optimization attempts to recover a candidate key $\boldsymbol{z}'$ using the recovery strategy described in §6.3.2. This search is initialized from a random $\boldsymbol{z} \sim \mathcal{Z}$ that is iteratively updated to optimize the recovery objective, attempting to identify a valid key that activates private behavior. This procedure is abstracted in Algorithm D.3 by the call to OptimizeKeyCandidate (line 20). Moreover, the recovery objective defined in §6.3.2 optimizes both a key and permutation matrix over the full dataset $\mathcal{D}$. Executing this procedure exactly during training would be infeasible, as it would require repeatedly iterating over the entire dataset within each outer training step. To make this stage tractable, we limit the inner optimization to a small number of steps S_{rob}. This truncated search produces a single candidate key $\boldsymbol{z}'$ intended to approximate the behavior of a recovery attacker. By keeping S_{rob} small, the additional cost becomes comparable to that of adversarial training methods, where inputs are iteratively perturbed within each batch.

The resulting candidate key is then applied to the current batch (line 20), and the preprocessor is trained to maintain public-mode behavior under this adversarial conditioning. Concretely, the optimization enforces that the preprocessor continues to implement the identity transformation (by minimizing the reconstruction norm) while preserving correct classification under the ground-truth labels (line 21). This effectively inverts the recovery objective described in §6.3.2, discouraging keys that would activate private behavior. The coefficient c_{rob} balances the reconstruction and classification terms in this objective.

Discussion: One may wonder why training against only a small number of candidate keys, each optimized for a few steps, suffices to harden against recovery in a high-entropy key space. The goal of the robustness objective is not to enumerate all possible aliases, but to eliminate *search-accessible* ones. Because public-mode training collapses randomly sampled non-secret keys to a single behavior while the secret key is explicitly associated with the private behavior, remaining aliases are likely to arise from *partial signatures* of the secret key—similarities in subregions of the key or in the internal activations they induce. In other words, the preprocessor may inadvertently associate private behavior with localized patterns in the key or its activations rather than with the key as a whole. While the public objective encourages isolation of the secret key, it does not strictly enforce collective matching, and the Monte-Carlo sampling used during training cannot fully cover the high-entropy space (see §6.3.2). Consequently, the small-measure assumption for the private region (discussed in §6.3.2) may be violated, allowing search or optimization procedures to exploit locally reachable signals that guide them toward nearby alias keys.

Robustness training suppresses these shared cues by forcing the model to respond to the secret key **as a whole** rather than to partial signatures of it. Once these signals are removed, any remaining aliases become effectively isolated and lack the locally exploitable structure required by search or optimization procedures; discovering them would therefore require near-random exploration in the high-entropy key space. The failure of diverse recovery attacks in §6.5.2, including gradient-based and gradient-free methods with varied initializations and substantially larger search budgets, supports this interpretation.

Algorithmic Details and Stage Transitions. Training progresses through stages once the objectives active in the current stage converge (lines 28–30). Algorithm D.3 abstracts this decision via the function CurrentStageSatisfied(), which evaluates epoch-level aggregated statistics of the active loss terms. In practice, a stage is considered converged when all active loss components stabilize across consecutive epochs—i.e., they no longer decrease meaningfully and do not increase. Once this condition is satisfied, the algorithm advances to the next stage in the curriculum described above. The ADV stage is skipped if the adversarial objective is disabled ($adv_obj = False$).

When the optional flag $early_termination$ is enabled (lines 22-23), an additional stopping condition is applied during the ROBUST stage based on aggregated batch-level statistics. Because robustness training involves an inner recovery optimization, continuing to process the entire dataset after convergence can be unnecessarily expensive for large datasets. In practice, we observe that the robustness objective typically converges rapidly. Therefore, when $early_termination$ is active, the convergence condition is evaluated at the batch level during the ROBUST stage. If CurrentStageSatisfied() holds, training terminates immediately and the preprocessor is returned. This allows robustness hardening to complete

after only a small number of batches when convergence occurs early, substantially reducing training cost for large-scale datasets.

The loss coefficients λ_{priv}, λ_{pub}, λ_{adv}, and λ_{rob} balance the different objectives in the training loss and are treated as hyperparameters—see Appendix D.2.3.

D.2 Experimental Setup Details

In this section we provide the details of our experimental setup. We first list the key space scale and dimension choices for the different datasets, and then describe the chosen U-Net backbones and preprocessor perturbation bounds. Afterward, we present the training configurations and hyperparameters. Finally, we discuss the setup and parameters of the adversarial attacks considered in §6.4.

D.2.1 Key Space Construction

The entropy of the key space is controlled by the scale σ and dimensionality d (see §6.3.2). We set $\sigma = 50.0$ to provide sufficient entropy to hide the private decision space. In §6.5.2 we contrast this choice with $\sigma = 1.0$, showing small scales make private behavior vulnerable to extraction.

For the dimensionality d, we match the input resolution (i.e., $d = H \times W$) of the dataset (32×32 for ***CIFAR*** and 224×224 for ***YouTube-Faces***). Larger key dimensions are unnecessary because the ConvolveCond operator rescales the key state to the current feature resolution at each conditioning point (Appendix D.1). Since U-Net feature maps are bounded by the input resolution, higher-dimensional keys would introduce redundant information that is discarded during this resizing step.

D.2.2 Preprocessor Configuration

Architecture. For all datasets the preprocessor is a U-Net augmented with ConvolveCond (Appendix D.1). The backbone depth is adapted to the datasets. Table D.1 summarizes the architectures used.

For ***CIFAR*** datasets, we employ a three-scale pyramid that compresses the feature resolution to 8×8 before expanding back to the input resolution. For ***YouTube-Faces*** (224×224 inputs) we use a shallower two-scale architecture, as the higher-resolution inputs

Dataset	**Input res.**	***z* dim (d)**	**# scales**	**Feature pyramid**
CIFAR-10	32×32	32×32	3	$32 \to 16 \to 8 \to 16 \to 32$
CIFAR-100	32×32	32×32	3	$32 \to 16 \to 8 \to 16 \to 32$
YouTube-Faces	224×224	224×224	2	$224 \to 112 \to 224$

Table D.1: U-Net backbone configurations.

already provide sufficient spatial structure and deeper pyramids offer limited benefit relative to their computational cost. For ***CIFAR*** we stack two U-Net modules, so the preprocessor effectively computes $U(\boldsymbol{x}, \boldsymbol{z}) = U_2(U_1(\boldsymbol{x}, \boldsymbol{z}))$, with the intermediate module propagating both the feature representation and key state. This increases the expressive capacity of the conditioning mechanism, allowing the network to focus on more complex key-specific patterns at lower key dimensionality. For ***YouTube-Faces***, a single U-Net backbone was sufficient due to the richer spatial structure of the learned higher-dimensional key states, which provides a stronger signal for associating the key with unique activation patterns that induce the private behavior.

Preprocessor Perturbation Budget ϵ. The preprocessor predicts a bounded perturbation $P(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x} + \tanh(U(\boldsymbol{x}, \boldsymbol{z}))\epsilon$, where ϵ controls the maximum transformation magnitude. Bounding the perturbation stabilizes training in multi-class settings by preventing large class-to-class interference (see §6.3.2).

Because inputs are normalized to $[0, 1]$, ϵ must be large enough to realize the label permutation while preserving sufficient input structure for input-adaptive transformations. Empirically, values above ≈ 0.125 allow reliable convergence with convergence occurring at a faster rate as ϵ grows as it increases the preprocessor's expressiveness. Yet, excessively large values destabilize optimization. We thus select ϵ based on early training dynamics. We use $\epsilon = 64/255$ for ***CIFAR-10/100*** and $\epsilon = 160/255$ for ***YouTube-Faces***, where the higher-resolution task benefits from a larger perturbation budget, which substantially accelerates convergence while preserving stable training.

D.2.3 Training Configuration

For each dataset, the preprocessor is trained using the curriculum described in §6.3.2 (see Appendix D.1.1).

Losses & Coefficients. For ***CIFAR-10*** and ***CIFAR-100***, we use the cross-entropy

loss for classification terms. For ***YouTube-Faces*** we use the Margin Logit Loss (*MLL*), which was more stable for this large-class regime. The loss coefficients balance the different objectives during training. We fix $\lambda_{\text{priv}} = 1.0$ for all datasets and calibrate the remaining coefficients relative to this reference. The resulting values used in our experiments are reported in Table D.2. The coefficients were selected through light validation tuning to ensure balanced optimization across objectives.

The constant c_{rob} balances the reconstruction and classification terms in the robustness objective—see Algorithm D.3. Because the goal of this stage is to enforce public-mode behavior for recovered keys, the reconstruction loss is the primary component, mirroring the public-collapse objective. To verify that reconstruction alone suffices, we set the classification term to a negligible weight $c_{\text{rob}} = 2 \times 10^{-5}$ for ***CIFAR-10*** and ***CIFAR-100***. Robustness was achieved without increasing this coefficient, indicating that the reconstruction loss alone suppresses key-search signals. Consequently, for ***YouTube-Faces*** we set $c_{\text{rob}} = 0$ and rely solely on the reconstruction term.

Dataset	λ_{pub}	λ_{adv}	λ_{rob}
CIFAR-10	10^3	10^{-1}	50
CIFAR-100	10^5	10^{-2}	5000
YouTube-Faces	10^2	–	100

Table D.2: Training loss coefficients. $\lambda_{\text{priv}} = 1.0$ for all datasets.

Adversarial Stage. The adversarial objective is applied for ***CIFAR-10*** and ***CIFAR-100*** whenever transfer attacks reduce private-mode robustness below that of prior baselines [207, 257, 14]—see Appendix D.1.3. In practice this condition was triggered for both datasets, and the stage was enabled using *PGD* with 50 iterations. For ***YouTube-Faces***, which serves as a demonstration dataset without comparable robustness baselines, transfer attacks did not significantly degrade private-mode accuracy and the adversarial stage was therefore omitted.

Robustness Training Parameters. During the robustness stage, the inner key-recovery search operates on mini-batches of training samples. For ***CIFAR-10*** and ***CIFAR-100***, the inner batch size is 256 and the search runs for $S_{\text{rob}} = 100$ steps. For ***YouTube-Faces***, the inner batch size is 32 and the search runs for $S_{\text{rob}} = 800$ steps, yielding a comparable effective sample budget across datasets. It is important to note that this truncated inner search is substantially weaker than a full recovery attack—see §6.5.2.

Training Procedure. We use the AdamW optimizer with an initial learning rate of 10^{-4} and apply a decay factor of 10 during the PUBLIC stage (only) whenever convergence stalls. If further decay does not improve optimization, the last decay is reverted and training proceeds to later stages without additional learning-rate scheduling. Because

additional curriculum stages introduce extra forward passes, batch sizes are reduced as new objectives become active. For ***CIFAR-10*** and ***CIFAR-100***, the batch size is 256 during the PRIVATE stage, 128 when the PUBLIC stage is added, and 64 for the remaining stages. For ***YouTube-Faces***, we use batch sizes 32, 16, and 8 respectively due to the higher input resolution.

For ***CIFAR-10***, training requires 89 epochs, with the costly robustness stage converging within two epochs only. For datasets with large class counts, early training can be unstable due to limited class coverage. Hence, we employ a short slow-start phase that initializes the private objective before the full curriculum. For ***CIFAR-100***, the small number of representatives per class complicates learning bounded input-adaptive perturbations that realize the private mapping through the pretrained classifier. We thus use 1M additional samples from [207] for 7 epochs to increase exposure, followed by 97 curriculum training epochs without these samples (the robustness stage requires 3 epochs). For ***YouTube-Faces***, the challenge stems from the vast number of classes (1283) relative to the batch size, which can lead to interference between batches. We initialize training using a subset containing a single sample per class for 94 epochs to establish a stable private mapping, followed by 20 curriculum epochs. For ***YouTube-Faces***, we employ the early-termination mechanism (see Appendix D.1.3), ending the robustness stage after roughly 100 batches within a single epoch.

D.2.4 Adversarial Attack Configuration

In our experiments we evaluate AA-ℓ_∞ [53], AA-ℓ_2 [53], PGD-ℓ_∞ [141], the Perceptual PGD attack ($PPGD$) [143], and the Low-Frequency attack (LF) introduced in DiffBreak [131] (see §5.5).

Implementation. All attacks are implemented using our *DiffBreak* library[1] from Chapter 5. *DiffBreak* integrates the multi-targeted AA implementation used in *RobustBench* [54] while providing correct gradient computation for diffusion-based purification (DBP) defenses such as *DiffPure*. As shown in Chapter 5, these corrected gradients reveal previously unobserved vulnerabilities in DBP defenses. Using this toolkit, therefore, allows for fair comparison with *RobustBench*-evaluated baselines, while ensuring accurate evaluation of DBP methods. Additionally, *DiffBreak* provides a unified interface for evaluating various attacks.

Hyperparameters. For ℓ_∞ attacks we use the common perturbation budget $\epsilon = 8/255$ for all datasets. For AA-ℓ_2 we use $\epsilon = 0.5$, a common setting in prior evaluations [131]

[1] https://github.com/andrekassis/DiffBreak

(this attack is not evaluated on ***YouTube-Faces***). For the *PPGD* attack we use an *LPIPS* constraint of 0.5 across all datasets, consistent with the settings used for ***CIFAR-10*** and ***ImageNet*** in [143] and the *DiffBreak* defaults. For the *LF* attack we follow the same configuration from §5.5.1. On ***CIFAR*** datasets we use the original *LPIPS* bound 0.05. For ***YouTube-Faces***, which was not evaluated in §5.5.1, we calibrate the threshold to avoid visible artifacts. A binary search yields a bound of 0.005, which preserves perceptual similarity while collapsing undefended classifiers (e.g., public-mode *KEYSHIFT* drops to 2.15%). For ***CIFAR-10*** and ***CIFAR-100***, the *LF* attack uses the adversarial filter network proposed in Appendix C.7.1. For ***YouTube-Faces*** we use the filter network instantiated for ***ImageNet*** in the same work section, which is better suited to higher resolutions.

KEYSHIFT Attack Strategy. Note that in the main experiments we mount attacks against *KEYSHIFT* in public mode and evaluate the resulting *AE*s in private mode—see §6.4. Public-mode attacks require a non-secret key to activate the preprocessor. To avoid dependence on a specific choice, at each attack step we sample 10 non-secret keys and perform an EOT update using the expected gradient. In practice, this produced results almost indistinguishable from using a fixed non-secret key or attacking the base classifier directly, as all non-secret keys collapse the preprocessor to the identity function.

DiffPure* Baseline.** When evaluating the *DiffPure* baseline, we follow the strict majority-vote (MV) protocol we propose in *DiffBreak* [131]—see §5.3.3, using $K = 10$ purified copies to reduce stochastic variance and increase its robustness, allowing for a fair comparison. Due to the computational cost of *DiffPure*, we evaluate it only under *LF* and *AA*-ℓ_∞ on ***CIFAR-10, as in §5.4.

D.3 Blackbox Attacks

Query-Based Attacks

Here, we provide detailed results for the *Square* attack experiments described in §6.5.1.

Results. Each *Square* iteration requires a single query. We thus treat 20 queries as a conservative budget consistent with security-critical operational constraints, and report robustness as a function of the query budget to illustrate how it degrades as attackers gain more interaction (Table D.3). With 20 queries, accuracy remains high at 73.63% on ***CIFAR-10*** and 40.92% on ***CIFAR-100***. In contrast, the strongest white-box attack

(AA-ℓ_∞) generating AEs in public mode to compromise private-mode *KEYSHIFT* from §6.4 attains lower robust accuracies (65.23% and 34.38% for ***CIFAR-10*** and ***CIFAR-100***, respectively).

These results indicate that *KEYSHIFT* maintains strong robustness against query-efficient black-box optimization under realistic limits. With 50–200 queries, robustness degrades gradually but remains substantial, reaching 43.55% and 19.34% after 200 queries. Naturally, increasing the query budget indefinitely would eventually entirely collapse robustness, which we verified. However, such attacks are out of scope as they violate the threat model, which operationally limits the allowed queries. Thus, as we view 20 queries as a permissive budget and since *Square* relaxes the threat model compared to viable, hard-label attacks. The results demonstrate that *KEYSHIFT* can protect classifiers in security-critical deployments that enforce the required restrictions.

Dataset	20 iters	50 iters	100 iters	200 iters
CIFAR-10	73.63	65.92	54.69	43.55
CIFAR-100	40.92	33.30	26.46	19.34

Table D.3: Private-mode robust accuracy (%) under the *Square* attack with varying query budgets.

Surrogate-Based Attacks

Another class of blackbox attacks relies on transferability: a surrogate model is trained to approximate the target model's decision boundaries and AEs are crafted against it and transferred to attack the target. Hence, we additionally evaluate *KEYSHIFT* against the Jacobian-Based Dataset Augmentation (*JBDA*) black-box attack [196]. *JBDA* queries the target model (oracle) to augment a synthetic dataset for training a surrogate model aligned with the target's decision boundaries.

Attack Description. Concretely, the attack assumes the adversary can obtain a small initial unlabeled set of samples, which they then expand by querying the target model iteratively. At each round, the oracle is queried on the current training set to obtain hard labels $\tilde{O}(\boldsymbol{x})$ for all unlabeled samples in this set. Then, the surrogate is trained on these samples. Afterward, new synthetic samples are generated as $\boldsymbol{x} + \lambda \cdot \text{sgn}(\boldsymbol{J}_F[\tilde{O}(\boldsymbol{x})])$ for each $\boldsymbol{x}$ in the current dataset, where $\boldsymbol{J}_F$ is the surrogate's Jacobian w.r.t. the oracle-assigned label. The process is then repeated for $\rho_{\max}$ rounds, each obtaining the labels for the newly-generated samples by querying the target oracle and thereby doubling the dataset size before retraining the surrogate. Intuitively, this augmentation strategy uses the surrogate's

Jacobian to identify its output-variation directions and then aligns these directions with the target's decisions to accurately model its boundaries.

Experimental Setup. We use a WRN-28-10 architecture for the surrogates, matching that of the *KEYSHIFT*-protected classifier (see §6.4.1), to promote surrogate-target alignment and potentially increase the attack's potency. Following the ***GTSRB*** configuration of [196] (the closest available to ***CIFAR-10*** in dimensionality), we train three independent ***CIFAR-10*** surrogates, each initialized with 1,000 samples drawn uniformly from the ***CIFAR-10*** test set (100 per class). We set $\lambda = 0.1$ and $\rho_{\max}$=6 augmentation rounds (further rounds were shown to be of diminishing returns [196])—yielding a final dataset of 32,000 samples labeled by querying private-mode *KEYSHIFT* (note that *KEYSHIFT* applies the inverse permutation before returning labels, preserving its secrecy). During each round, the surrogate is trained for 100 iterations on the current set and all three surrogates achieve near-perfect label agreement with the target system on their training sets. *AE*s are then crafted on each surrogate using the five attacks from §6.4 and evaluated for transferability to private-mode *KEYSHIFT*.

This *JBDA* query budget already far exceeds realistic security-critical deployment constraints under our threat model (see §6.2). Account creation typically also requires identity validation, precluding sybil-style query farming. We nonetheless treat *JBDA* as a *conservative upper bound* on a realistic static surrogate attack, where an attacker obtains samples previously labeled by the target model via a data breach to train an accurate surrogate. In the considered environments, safeguards are typically deployed to limit the scale of such breaches, making the dataset size we consider a realistic approximation. Yet, such leaked samples would be passively collected and thus less boundary-informative than *JBDA*'s actively selected ones, making *JBDA* a more impactful attack.

Attack	**Surr. 1**	**Surr. 2**	**Surr. 3**	**Mean ± Std**
AA-ℓ_∞	74.02	75.10	73.14	74.09 ± 0.98
AA-ℓ_2	82.81	83.50	81.93	82.75 ± 0.79
PGD-ℓ_∞	78.32	78.32	77.05	77.90 ± 0.73
LF	74.12	74.71	73.14	73.99 ± 0.80
PPGD	79.49	80.47	80.37	80.11 ± 0.54

Table D.4: Robust accuracy (%) of private-mode *KEYSHIFT* on ***CIFAR-10*** under transferred *AE*s generated against *JBDA* surrogates.

Results. As shown in Table D.4, robust accuracy remains consistently high across all surrogates and attack types, with low variance across the three surrogates confirming the stability of these findings. The surrogate-based results are strictly higher than those of white-box adversaries who craft *AE*s in public mode targeting private-mode *KEYSHIFT* (see §6.4),

with gaps ranging from +4.07% (PGD-ℓ_∞) to +8.86% (AA-ℓ_∞). This demonstrates that under realistic black-box access, attackers fail to build surrogates that provide any advantage over white-box adversaries lacking *KEYSHIFT*'s secret parameters. While surrogate-based adversaries gain some insight into the model owner's decision process (labels), the absence of gradients and query-budget constraints prevent them from accurately capturing the decision boundaries. In contrast, the white-box adversaries of §6.4 lack access to the private decision process but exploit the shared architecture and weights between public and private modes, enabling some transferability. All in all, *KEYSHIFT* remains robust to both attack types.

Analysis. Accurately modeling a target's decision boundaries requires identifying the directions that cause the target itself to vary its outputs and designing a surrogate that follows these directions closely. Yet, without access to the target's internals—such as gradients, which are unavailable in private mode and infeasible to estimate under query-rate limits—such directions cannot be directly recovered. *JBDA* addresses this by using a proxy: it identifies output-variation directions of the **surrogate** instead and reinforces or eliminates them based on their agreement with the target's decisions, attempting to approximate the desired alignment more efficiently. While effective in many settings [196], this heuristic loses its efficacy against *KEYSHIFT*, wherein the preprocessor manipulates the protected, pretrained classifier's decisions through specialized input-adaptive transformations, significantly increasing the complexity of the decision space such that surrogate-guided directional agreement alone no longer suffices to capture the target's true boundaries.

Summary

Together, these results indicate that *KEYSHIFT* retains strong robustness against black-box adversaries under realistic operational constraints. Query-based optimization fails to reliably discover adversarial directions within the limited interaction budgets typical of security-critical deployments, while surrogate-based attacks cannot accurately approximate the protected decision process. Consequently, without access to the secret key or gradients of the private mode, black-box adversaries gain no practical advantage over the already considered white-box transfer attacks.

D.4 Complexity Discussion

Inference Cost. At inference time, *KEYSHIFT* introduces a single forward pass through the preprocessor before classification. Since the preprocessor is merely a U-Net, the inference

cost is comparable to adding one lightweight image-to-image network before the classifier. In practice, this introduces only modest latency relative to the underlying classifier.

Training Cost. Training the preprocessor is more computationally demanding than inference. The cost arises from three components: (i) optimization of the U-Net backbone, (ii) adversarial training when the optional adversarial objective is enabled, and (iii) the robustness stage, which introduces an inner key-recovery optimization loop at each training step. The latter is the dominant cost per iteration because it requires repeated forward and backward passes during the inner search.

On ***CIFAR-10***, the first two stages of training run at approximately 1.1 batches/s (196 batches per epoch for the private objective and 391 batches per epoch for the joint private/public stage). The robustness stage is substantially more expensive, requiring roughly 20 seconds per batch (782 batches per epoch). However, this stage typically terminates early (within $\sim$ 2 epochs). Across the full curriculum ($\sim$ 89 epochs), the majority of training thus occurs in the faster stages, resulting in total training times of well under one day on a single A100 GPU.

For the larger-scale ***YouTube-Faces*** setting, total training required roughly one week on the same hardware. The increase is driven primarily by the much larger number of training batches per epoch and higher dimensionality, although the expensive robustness stage terminates early. This dataset involves 224×224 resolution inputs and 1,283 classes with over 370K training samples, making it representative of large-scale workloads. The resulting training times demonstrate that *KEYSHIFT* can be trained at scales relevant to real-world applications.

Scalability. The training procedure is readily parallelizable. Both the outer optimization and the inner search operate on mini-batches and can therefore be distributed across multiple GPUs using standard data-parallel training. The robustness stage search steps are also naturally parallelizable. Consequently, organizations deploying *KEYSHIFT* can significantly reduce training time using multi-GPU infrastructures.