Towards Honest, Practicable and Efficient Private Learning

by

Shubhankar Mohapatra

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, 2025

© Shubhankar Mohapatra 2025

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:	Shahab Asoodeh Assistant Professor, Dept. of Computing and Software, McMaster University
Supervisor:	Xi He Associate Professor, Computer Science, University of Waterloo
Internal Member:	Gautam C. Kamath Assistant Professor, Computer Science, University of Waterloo
Internal-External Member:	Helen Chen Professor of Practice, Public Health Sciences, University of Waterloo
Other Member(s):	Sujaya Maiyya Assistant Professor, Computer Science, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this 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.

Abstract

Protecting our personal information is a major challenge in today's data-driven world. When scientists and companies analyze large datasets, they need a way to ensure our individual privacy isn't compromised. This thesis focuses on Differential Privacy, a powerful, mathematical guarantee that places a strict, verifiable limit on how much personal information can be leaked, even if an attacker has the worst-case advantage. Researchers have developed various sophisticated algorithms to accomplish useful tasks, like building machine learning models or generating realistic synthetic data, while maintaining Differential Privacy. Crucially, these operations must be conducted within a predetermined, strict limit, often referred to as the "privacy budget." This budget mathematically quantifies the total acceptable loss of privacy for the entire process, enforcing a crucial trade-off between data utility and individual protection. All routine procedures of the machine learning pipeline, including data cleaning, hyperparameter tuning, and model training, must be performed within the budget. Several tools can perform these tasks in disjunction when the dataset is non-private. However, these tools do not translate easily to differential privacy and often do not consider the cumulative privacy costs. In this thesis, we explore various pragmatic problems that a data science practitioner may face when deploying a differentially private learning framework from data collection to model training. In particular, we are interested in real-world data quality problems, such as missing data, inconsistent data, and incorrectly labeled data, as well as machine learning pipeline requirements, including hyperparameter tuning. We envision building a general-purpose private learning framework that can handle real data as input and can be used in learning tasks such as generating a highly accurate private machine learning model or creating a synthetic version of the dataset with end-to-end differential privacy guarantees. We envision our work will make differentially private learning more accessible to data science practitioners and easily deployable in day-to-day applications.

Acknowledgements

My PhD journey has been a remarkable adventure. I am immensely grateful to have had this opportunity in life, and I owe it to many people.

I want to begin by thanking my supervisor, Professor Xi He. She has been a guiding light through this journey and has taught me how to be an honest, hardworking, and truthful researcher. I am also grateful to Professor. Gautam Kamath, who has also stood by me and helped me navigate through the various thick and thin of this journey. I have also had the opportunity to work with many brilliant people who have shaped this thesis in the best possible ways. I am grateful to each of them and my examining committee for providing me with helpful feedback and comments that have helped improve my work.

It is well known that the life of an international student can be challenging in many ways. However, I am blessed to have chosen Canada and the University of Waterloo for this journey. I have received a lot of love and support from the people here. I am also grateful to the Ontario government for awarding me the Ontario Graduate Scholarship twice during my graduate studies, and to the School of Computer Science for awarding me the Cheriton Graduate Scholarship, which supported my research.

I have also been fortunate to have made some fantastic friends during my grad studies. I carry with me numerous memories of board game nights, parties, hikes, camping, and sports events, many of which took place at the university. I am thankful to them for these memories that got me through the tough times. I was also blessed to have roommates who were no less than family to me. A significant portion of my PhD was during the COVID-19 pandemic, and we had to spend time at home due to travel restrictions. These were the times when we looked after each other and survived through the tough times. I would like to extend a heartfelt thank you to all the roommates I have had during my time at Waterloo: Sagar, Teja, Manav, Aseem, Rishav, Sahu, Khadija, Boga, Ashirbad, Ana, and Aaishwarya.

Last but not least, I am grateful to my parents, brother, and all the people who have been rooting for me back in India. Without your support, sacrifices, and love, I would not have been able to complete this thesis.

Thank you so much! Your support and blessings mean the world to me!

Dedication

This thesis is wholeheartedly dedicated to my parents, Santosh and Lilima.

Table of Contents

Examining Committee ii

Author's Declaration iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures x

List of Tables xiii

1 Introduction 1

1.1 Problem Setup 3

1.2 Thesis Contributions 4

1.2.1 Computing Inconsistency Measures under Differential Privacy . . . 5

1.2.2 Generating Synthetic Data with Missing Data 6

1.2.3 Honest Hyperparameter Tuning using Adaptive Optimizers 8

1.3 Thesis Outline 9

2 Background **10**

2.1 Notations . . . 10

2.2 Differential Privacy . . . 10

2.2.1 Definition . . . 11

2.2.2 Algorithms for DP . . . 12

2.2.3 Properties of DP . . . 13

2.2.4 DP for Graphs . . . 14

2.2.5 DP for Machine Learning . . . 15

2.2.6 DP for Synthetic Data Generation . . . 16

3 Data Exploration and Cleaning: Computing Inconsistency Measures **18**

3.1 Introduction . . . 18

3.2 Preliminaries . . . 20

3.2.1 Database Constraints . . . 20

3.2.2 Inconsistency Measures . . . 21

3.3 Inconsistency Measures under DP . . . 23

3.4 DP graph projection for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ . . . 25

3.4.1 Graph Projection Approach for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ . . . 25

3.4.2 Optimized Parameter Selection . . . 32

3.5 DP Minimum Vertex Cover for $\mathcal{I}_{\mathsf{R}}$ . . . 37

3.6 Experiments . . . 42

3.6.1 Experimental Setup . . . 42

3.6.2 Results . . . 44

4 Data exploration and Cleaning: Generating synthetic data with missing data **51**

4.1 Introduction . . . 51

4.2 Preliminaries on Missing Data . . . 52

4.3 Problem Statement . . . 54
4.4 Privacy for Incomplete Data . . . 55
4.4.1 Vanilla Approaches . . . 55
4.4.2 Adaptive Recourse Approach . . . 58
4.5 Privacy for Groundtruth Data . . . 66
4.5.1 Relationship to Problem 1 . . . 66
4.5.2 Privacy Amplification Due To MCAR . . . 68
4.6 Evaluation . . . 75
4.6.1 Experimental Setup . . . 76
4.6.2 Experimental Findings . . . 79

5 Hyperparameter Tuning: Honest Tuning using Adaptive Optimizers 84
5.1 Introduction . . . 84
5.2 Preliminaries for DP Optimizers . . . 85
5.3 Problem Setup and Overview . . . 86
5.4 Privately Tuning DP Optimizers . . . 87
5.5 Tuning DP Optimizers . . . 91
5.6 Advantages of Tuning using DPAdam . . . 96
5.7 DPAdamWOSM . . . 99

6 Related Work 102
6.1 Differential Privacy and Data Repair . . . 102
6.2 Differentially Private Synthetic Data Generation for Missing Values . . . 103
6.3 Hyperparameter Tuning with Differential Privacy . . . 104

7 Conclusion 105
7.1 Contributions . . . 105
7.2 Future Work . . . 106

References 108

List of Figures

1.1 An illustration of the workflow in the envisioned framework. Each step of the learning pipeline (data exploration, cleaning, hyperparameter tuning and model training) is privatized using part of the privacy budget ε to generate the output based on the requirements of the data curator. 3

3.1 Toy example dataset to show a worst-case analysis. An additional row may violate all other rows in the dataset (left). Easier analysis can be done by instead converting the dataset into its corresponding conflict graph (right). 21

3.2 Toy graph example $\mathcal{G}$ with seven nodes (A to G) and seven edges. Consider a neighboring graph $\mathcal{G}'$ with the differing node E (red) and its two edges. . 37

3.3 True vs Private estimates for all datasets with RNoise $\alpha = 0.01$ at $\varepsilon = 1$. Each row shows the three measures — $\mathcal{I}_{\mathsf{P}}$ (Positive degree nodes), $\mathcal{I}_{\mathsf{MI}}$ (Number of edges), and $\mathcal{I}_{\mathsf{R}}$ (Size of vertex cover) — for the corresponding dataset. 45

3.4 True vs Private estimates for all datasets with RNoise $\alpha = 0.01$ at $\varepsilon = 1$. Each row shows the three measures — $\mathcal{I}_{\mathsf{MI}}$ (Number of edges), $\mathcal{I}_{\mathsf{P}}$ (Positive degree nodes), and $\mathcal{I}_{\mathsf{R}}$ (Size of vertex cover) — for the corresponding dataset. 46

3.5 Computing different strategies for choosing θ for all datasets with RNoise at $\alpha = 0.01$ and $\varepsilon = 1$. The datasets are arranged according to their densities from sparsest (top) to densest (bottom). 47

3.6 Computing inconsistency measures for different datasets with RNoise at $\alpha = 0.005$ and varying privacy budgets 50

4.1 Complete row only approach results in poor results for MAR and MNAR missing mechanism. The vertical lines denote the std dev. across 3 runs. . 56

4.2 Illustration of worst case statistical imputations. 59

4.3 Example of private incomplete datasets generated from two neighbouring ground truth datasets. Gray denotes missing cells and dotted lines represent the differing row. 67

4.4 The effect of missing data on DP synthetic data generation algorithms. . . 76

4.5 Comparing all strategies to deal with missing data. Adaptive recourse results in the best performance, followed by the complete row approach. The red line represents the best no-missing baseline. 77

4.6 Adaptive methods (DPMisGAN, PrivBayesE, KaminoI) vs their respective baselines using the complete row approach (DPautoGAN, PrivBayes, Kamino) with MCAR missingness at $\varepsilon = 1$. Algorithms of the same category are colored with the same shade. The adaptive methods result in better quality synthetic data. The red line denotes the best no missing baseline. . 78

4.7 Adaptive methods vs non-adaptive methods on different missing mechanisms. The red line denotes the no missing baseline. 80

4.8 Baselines vs Adaptive methods at $\varepsilon = 3$. The adaptive methods perform better than their respective baselines. 81

4.9 Varying privacy budget and comparing utilities of our approaches. 82

5.1 Comparing the privacy cost of LT versus Moments Accountant. The minimal privacy overhead incurred by LT is at least ≈5x, and increases with the dataset size (left). However, as we allow LT to sample and test more candidate hyperparameters, the privacy cost barely increases (middle). Moments Accountant is able to test a significant number of candidates at the same cost as the minimal privacy overhead of LT (right). 89

5.2 Comparing LT vs MA with varying number of candidates at setting $\sigma = 4$, $L = 250$, $T = 10000$. MA can compose up to 14 and 26 candidates for the same cost of LT for dataset sizes 5k (left) and 60k (right), respectively. . . 90

5.3 Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPSGD. The white pixels correspond to training losses below the 1st percentile. Note that most best loss values lie on a diagonal expressing the inverse connection between α and C. 93

5.4 Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPAdam. The white pixels correspond to training loss lower than 1 percentile of DPSGD counterpart. Note that best loss values are dependent on α and spread over a wider range of C. 93

5.5 Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPSGD. The white pixels correspond to training losses below the 1st percentile. Note that most best loss values lie on a diagonal expressing the inverse connection between α and C. 94

5.6 Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPAdam. The white pixels correspond to training loss lower than 1 percentile of DPSGD counterpart. Note that best loss values are dependent on α and spread over a wider range of C. 94

5.7 Ranking hyperparameter candidates across datasets. The black points correspond to the candidates with $\alpha = 0.001$ (with all permutations of β_1, β_2 from our search grid); the gold corresponds to the candidate with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$. 95

5.8 Comparing the testing accuracy curves of DPAdam, DPSGD and DPMomentum models across their hyperparameter tuning grids with $\sigma = 4$. The limits for y-axis are adjusted based on the dataset while maintaining a 15% range for all. 97

5.9 Comparing the testing accuracy curves of DPAdam, ADADP and DPAdamWOSM models across the hyperparameter tuning grid from Table 5.2 with $\sigma = 4$. The y-axes limits are adjusted based on the dataset while maintaining a 15% range for all. 99

List of Tables

2.1 Table of notations 11

3.1 Relative errors for a SQL approach vs our approach to compute the minimal inconsistency measure at $\varepsilon = 1$ 20

3.2 Summary of Inconsistency Measures, $n = |D| = |\mathcal{G}_\Sigma^D.V|$, $m = |\mathcal{G}_\Sigma^D.E|$, Θ is the candidate set. 22

3.3 Quality function computation for $\mathcal{I}_{\mathsf{MI}}$ for the conflict graph in Figure 3.1 when $\varepsilon_2 = 1$ 27

3.4 Probabilities of choosing candidates with exponential mechanism (EM) vs two-step hierarchical exponential mechanism (2-EM). θ_1^* refers to the output of the first step of 2-EM. 35

3.5 Description of datasets. The max deg column shows the maximum degree of any node in a $10k$ rows subset of the conflict graph of the dataset with 1% RNoise. 42

4.1 Dataset Characteristics 76

4.2 Amplified privacy for ground truth data. 83

5.1 Datasets used in experiment 96

5.2 Parameter grid for comparing DPSGD and DPAdam 98

5.3 Final ε (at $\delta = 10^{-6}$) for optimizers for the LR Models (Figure 5.8). DPSGD and DPMomentum use LT for privacy accounting; DPAdam uses MA. . . . 98

Chapter 1

Introduction

"Privacy is not an option, and it shouldn't be the price we accept for just getting on the Internet."

— Gary Kovacs

Each passing minute of the modern world now stores petabytes of data, and our human race strives to store and analyze this data [1]. Learning from this data has led us to the advent of many technological artificial intelligence (AI) advancements, such as personal assistants, self-driving cars, and futuristic humanoids. We have realized that most modern AI applications improve when we feed higher quality and larger amounts of data [161]. However, most of the required data in actual applications is often attributed to personal information(e.g., medical, financial, and social media data). If proper care is not taken while training, AI applications may lead to the undesirable disclosure of sensitive information [15, 80, 83]. Consider the following example from a medical research institute to predict diabetes.

Example 1. *A medical research institute aims to predict the onset of diabetes based on an individual's daily habits. To explore the relationship between your current health vitals and practices and the possibility of being diagnosed with diabetes, the research institute conducts a questionnaire across various demographics. The study training dataset included sensitive information such as smoking and eating habits. To prevent data leakage, the medical research institute anonymizes the data by removing all identifiable information. Due to its own inability to train ML models or lack of compute, the research institute then assigns a third-party company to build an AI model to input a person's vital health signs and predict the disease's outcome. Later, the model was released to the public so anyone could*

check their private prediction. The model did not release your data directly. However, an insurance company could use the AI model to infer that your data was involved in training the particular model and increase your life insurance premium.

Such incidents involving private information have been reported in multiple instances [132, 83]. In 2008, Netflix had released an anonymized dataset about their users for a public movie recommendation system building competition. However, researchers were able to identify individuals in the dataset by extrapolating information in the dataset with publicly available information [132]. More recently, in 2023, a Korean dating company released a chatbot trained on user information. This chatbot soon started to output text verbatim from user's sensitive dating information [83]. In both these incidents, both companies had to pay huge penalties as remuneration to the users for their data. Due to these mishaps, we now know that private data leakage can be dangerous and may lead to adverse effects. Furthermore, we have also learned that naive anonymization is not a viable solution to prevent private data leakage. Various papers have shown that malicious adversaries can even leverage carefully anonymized data to develop attacks that reveal the identity of individuals (membership inference) [159] or reconstruct sensitive training data by having access to the model's output (model inversion) [60]. Many regulatory policies, such as the General Data Protection Regulation (GDPR) [2] and the California Consumer Privacy Act (CCPA) [141], have been put forward to provide guidelines on educating people about their rights to personal data and penalize them with fines and legal actions for organizations that do not adhere to these regulations.

The computer science community has suggested multiple notions of privacy, but the one that has gained the most popularity is that of differential privacy (DP) [52]. It has emerged as the formal privacy guarantee with strong theoretical guarantees and has been adopted by various industry and government bodies [8, 56, 44]. Informally, DP enforces that the output of a data-sharing process has a similar distribution, regardless of whether an individual's data was used for the computation or not. A privacy budget denoted by ε is defined as one that controls the total privacy loss in a differentially private system. Although there are many ways to achieve DP, a standard process is to obscure the computation output by adding noise proportional to the privacy budget. In the last two decades, there has been flourishing research in differential privacy for its usefulness in various tasks in computer science, including those in databases [194, 195, 97, 196], machine learning [4, 43, 175], and generative AI [158, 67, 46]. Although these approaches work well for the task at hand, most assume that the model curator has unfettered access to the private dataset and that only the process output should be private. For example, prior works on training machine learning models [175, 4] propose algorithms to train differentially private machine models but do not account for hyperparameter tuning. They assume somehow that the

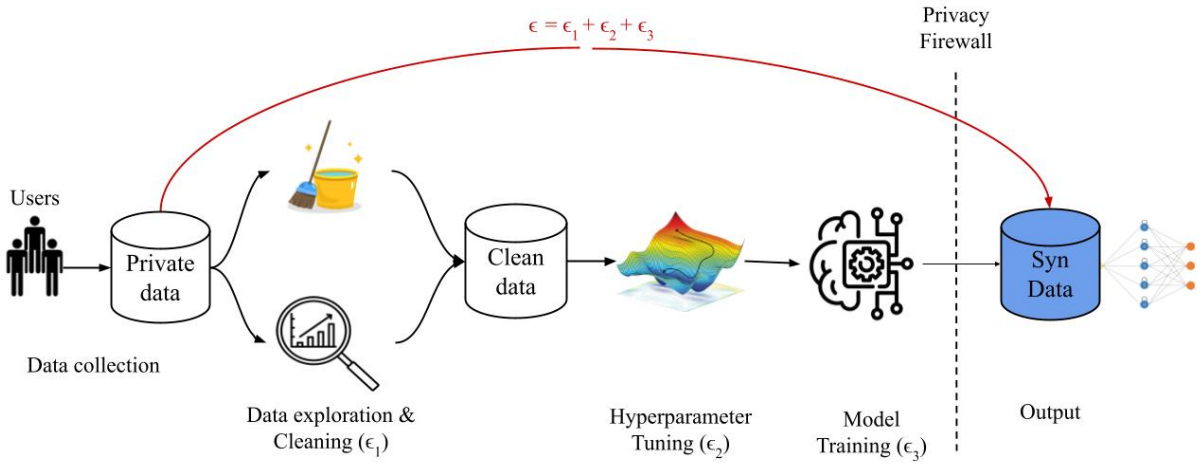


Figure 1.1: An illustration of the workflow in the envisioned framework. Each step of the learning pipeline (data exploration, cleaning, hyperparameter tuning and model training) is privatized using part of the privacy budget ε to generate the output based on the requirements of the data curator.

best hyperparameters were known ahead of time. Similarly, prior works for synthetic data generation [195, 158] assume that the input private data is error-free (i.e., free of missing values, integrity constraint violations, etc.). However, real-world data is often filled with errors in practice, and data cleaning is integral to learning. To tackle these problems, in this thesis, we propose, delineate challenges, and make progress towards building an end-to-end differentially private learning framework that can be used in practice for real-world applications.

1.1 Problem Setup

We consider a setup where a data curator wishes to release an ML model trained on a highly sensitive dataset similar to Example 1. They don't have the means to train the model and hence assign a third-party model curator (e.g. any AI company) for the job. However, they wish to prevent any leakage from the data by leveraging differential privacy with a preset strict privacy budget. In our work, we assume a strict setup where the the model curator is honest but curious and the model after training is released in the wild where any malicious adversary (e.g. an insurance company) could attempt to learn about the dataset. Training with differential privacy allows to defend against worst case adversaries who could have knowledge about all but one individual in the dataset.

The job of the model curator is hard as they need to create a private learning pipeline that is differentially private and judiciously uses the privacy budget to achieve the best model possible. In this thesis, we try to solve the model curator's problem by ideating an end-to-end differentially private learning pipeline as shown in Figure 1.1. The pipeline is based on a standard ML pipeline from data collection to final model training. It starts with the collected private data and involves three significant steps (data exploration and learning, hyperparameter tuning, and model training). Although we use the example of building an ML model, these steps remain mostly the same for other outputs, such as generating synthetic data or answering statistical questions about the data. Most prior works in the field have focused on the last part of the pipeline, i.e., model training, but the first two steps have been overlooked. Three significant challenges come up when we consider learning in this fashion:

1. *Limited privacy budget* - A differentially private framework consists of one privacy budget ε. Each step of the pipeline should be made differentially private using this budget.

2. *Cost of hyperparameter tuning* - Most works do not consider the cost of hyperparameter tuning. In practice, this step is done by linearly iterating over multiple hyperparameter candidates and choosing the best one. However, with privacy, the cost of hyperparameter tuning scales with every run.

3. *Presence of data inconsistencies* - Real-world data is often accompanied by inconsistencies such as missing values or incompatible formatting. These errors are often resolved by hand, but that gets difficult when the dataset is private.

1.2 Thesis Contributions

We propose three contributions towards this end-to-end differentially private learning pipeline. Two of our contributions fit in the data exploration and cleaning step, and a third in the hyperparameter tuning step. The code developed as part of this thesis is available on Github [124, 127, 125]. Next, we give an overview of the three contributions. For each, we provide a broad overview including how they fit into the end-to-end pipeline and the challenges we tackle.

1.2.1 Computing Inconsistency Measures under Differential Privacy

Overview The first challenge we tackle in this thesis is identifying whether a private dataset has any inconsistencies. Assessing inconsistencies/quality is important before investing money and effort into a specific dataset. Data quality is often impacted by inconsistency arising from diverse sources (encyclopedias, historical data, social networks, sensors, etc.), procedures (surveys, ML methods, signal processing, etc.), or the consolidation of multiple such sources and procedures. Prior research [170, 142, 114] has explored how to measure database inconsistency concerning *integrity constraints* (e.g., functional dependencies) and its implications for downstream analytics. Integrity constraints are a set of rules used in database management systems (DBMS) to ensure the accuracy, consistency, and reliability of data.

An elegant solution is to use *inconsistency measures* [170, 142, 114] that quantify data quality with a single metric for all constraints. This approach aligns well with DP, as such measures give a single numerical value representing data quality. Prior work describes various measures, such as the *problematic measure* (counting constraint tuples), the *minimal inconsistency measure* (counting violations), and the *repair measure* (minimal deletions for consistency). These measures often exhibit a broader range, making them potentially more suitable for estimating inconsistency under DP. In this paper, we delve into these three representative inconsistency measures, along with two additional measures identified in prior work [114], to assess their suitability for private datasets.

We formulate the problem of computing inconsistency measures as learning graph statistics with differential privacy [75, 92, 42]. We present algorithms that effectively estimate these measures for practical settings. This approach leverages the graph projection techniques from the state-of-the-art DP algorithms [42]. While these algorithms have proven effective in prior studies on social network graphs, they may encounter challenges with conflict graphs arising from integrity constraints. We provide a utility analysis highlighting these challenges for the general approach adapted from previous work and offer optimization techniques that prove effective for various conflict graph sizes and sparsity levels. In addition, for the optimal repair measure, an NP-hard problem, we introduce a novel 2-approximation algorithm with differential privacy guarantees. We also identified two of the five measures as incompatible with the DP setting.

Contributions The main contributions of this paper are as follows:

- We formulate the novel problem of computing inconsistency measurements with DP

for private datasets and discuss the associated challenges, including a thorough analysis of the sensitivity of each measure.

- We devise several algorithms that leverage the conflict graph and algorithms for releasing graph statistics under DP to estimate the measures that we have determined are suitable.
- We propose a new optimization for choosing graph truncation threshold that is tailored to conflict graphs and augment the classic vertex cover approximation algorithms to bound its sensitivity to 2 to obtain accurate estimates of the measures.
- We present experiments on five real-world datasets with varying sizes and densities to show that the proposed algorithms are efficient in practice.

1.2.2 Generating Synthetic Data with Missing Data

Overview In the second work, we look into the problem of having missing values in the private dataset when learning. For this work, we specifically focus on synthetic data generation as the output of the learning framework. Despite a number of work [193, 62, 146, 87] that succeed in generating synthetic data with DP guarantees, they only look at a simple scenario where the input data has no missing values. Several prior studies [107, 66] have reported on the prevalence of missing data in various fields. For instance, a study of 9 publicly available healthcare datasets commonly used in machine learning research found that the proportion of missing values ranged from 0.2% to 78.6% [66]. The presence of missing data can be attributed to multiple reasons, such as human errors [11] and privacy regulations like GDPR [2] which allow people the "right to forget" where one may ask their data to be deleted completely [148].

In our work, we ask how missing data will affect the quality of the synthetic data generated by mechanisms that offer DP guarantees. Our preliminary study shows that existing DP mechanisms have 4%-18.5% decay in the F1-score of downstream ML tasks on the synthetic dataset generated from a dataset with 10% missing values as compared to that when generated from the complete dataset. The decay varies depending on the types of missing mechanisms and the types of data generation processes.

We address the challenge of generating differentially private (DP) synthetic data from sensitive datasets with missing values. Our research explores two privacy guarantee scenarios: protecting the incomplete data or the ground truth data.

A unified solution is complex due to the lack of a universally optimal DP synthetic data generation algorithm, even without missing data, and the vast design space for handling missing data. We investigate various techniques, including using only complete rows, common imputation methods (statistical, machine learning), and DP imputation approaches. However, these methods often lead to limitations like data loss or high privacy costs, resulting in poor-quality synthetic data. To address this, we propose a comprehensive set of approaches, comprising two "vanilla" and three "adaptive" methods, to map out the design space for this new problem.

Furthermore, we examine the relationship between privacy guarantees for incomplete and ground truth data. By modeling the missing mechanism as a sampling process, we achieve a tighter upper bound for privacy loss to the ground truth data using sampling amplification techniques. Unlike previous work, our approach leverages the inherent randomness of missing values to enhance privacy for ground truth data.

Contributions The major contributions of our work are as follows:

- We are the first to formalize and study the problems of DP synthetic data with missing data. Our results show that existing algorithms have a decrease of 5-23% in utility with $\leq 5\%$ missing values and a decrease of 10-190% with $\leq 20\%$ missing values.
- We develop three novel adaptive approaches, each tailored to an existing category of DP mechanisms, that seamlessly combine dealing with missing data along with the learning process. Our evaluation shows that they improve the utility of the synthetic datasets by up to 15-72%. These simple yet effective approaches sometimes even achieve the same utility as the synthetic data generated from the no-missing ground truth data.
- We differentiate the privacy guarantees for both incomplete and ground truth data. Our analysis develops sufficient conditions that when satisfied, the algorithms for the incomplete data can be used to achieve privacy for the ground truth data.
- We are the first to apply amplification due to missing mechanisms and tighten the privacy bound for ground truth data. The amplified ground truth privacy is 0.1-0.65x the privacy achieved for the incomplete data with 10-50% missing values.

1.2.3 Honest Hyperparameter Tuning using Adaptive Optimizers

Overview DP training procedures have been developed [181, 20, 162, 5], which generally resemble non-private gradient-based methods, but with the incorporation of gradient clipping and noise injection. In ML training *hyperparameter selection* is instrumental to achieving high accuracy. The most common methods are grid search or random search, both of which incur a computational overhead scaling with the number of hyperparameters under consideration. In the private setting, this issue is often magnified as most private training procedures introduce new hyperparameters. Regardless, and more importantly, hyperparameter tuning on a sensitive dataset also costs in terms of *privacy*, naively incurring a multiplicative cost which scales as the square root of the number of candidates (based on composition properties of differential privacy [88]).

Most prior works on private learning choose not to account for this cost [5, 192, 176], focusing instead on demonstrating the accuracy achievable by private learning under idealized conditions; if the best hyperparameters were somehow known ahead of time. Some works assume the presence of supplementary public data resembling the sensitive dataset [16, 152], which may be freely used for hyperparameter tuning. Naturally, such public data may be scarce or nonexistent in settings where privacy is a concern, leaving practitioners with little guidance on how to choose hyperparameters in practice. As explored in this thesis, poor hyperparameter selection with standard private optimizers can have catastrophic effects on model accuracy.

Hope is afforded by the success of adaptive optimizers in the non-private setting. The canonical example is Adam [94], which exploits moments of the gradients to adaptively and dynamically determine the learning rate. It works out of the box in many cases, providing accuracy comparable with tuned SGD. However, Adam has been overlooked in the context of private learning since previous works have show than fine-tuned DPSGD tends to perform better than DPAdam [138], which has lead to several subsequent works [191, 176] to limit themselves to highlighting accuracy under ideal DPSGD hyperparameters. We navigate the different options available to a practitioner to solve the *honest private hyperparameter tuning problem* and ask, *are there optimizers which provide strong privacy, require minimal hyperparameter tuning, and perform competitively with tuned counterparts?*

Contributions The main contributions for this work are:

- We investigate techniques for private tuning of hyperparameters. We perform the first empirical evaluation of the proposed theoretical method of [109] and demon-

strate that it can be expensive; in certain cases, one can tune over sufficiently many hyperparameters using standard composition tools such as moments accountant [5].

- We empirically and theoretically demonstrate that two hyperparameters, the learning rate and clipping threshold, are intrinsically coupled for non-adaptive optimizers. While other hyperparameters and the model architecture are restricted by the scope of the task, privacy and utility targets, and computational resources, the learning rate and clipping norm have no a priori bounds. Since the resulting hyperparameter grid adds up to the privacy cost while tuning to achieve the model with the best utility, we explore leveraging adaptive optimizers to reduce the hyperparameter space.

- We empirically demonstrate the DPAdam optimizer (with default values for most hyperparameters), can match the performance of tuned non-adaptive optimizers on a variety of datasets, thus enabling private learning with honest hyperparameter selection. This finding complements a prior claim of [138], which suggests that a well-tuned DPSGD can outperform DPAdam. However, our findings show that this difference in performance is relatively insignificant. Furthermore, in the realistic setting where hyperparameter tuning must be accounted for in the privacy loss, we show that DPAdam is much more likely to produce non-catastrophic results.

- We show that the adaptive learning rate of DPAdam converges to a static value. To leverage this, we introduce a new private optimizer, DPAdamWOSM that matches the performance of DPAdam without computing the second moments.

1.3 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we present the necessary common preliminaries for our work. The next three Chapters 3, 4, 5 are dedicated to each main contribution of the thesis. The chapters contain an introduction to the problem, necessary preliminaries, the contribution, and experiments. Next, in Chapter 6, we present the related work for the thesis. Finally, the final chapter 7 concludes the thesis with final remarks and scope for future work.

Chapter 2

Background

This chapter provides the necessary common background and preliminaries for this thesis. Each chapter's specific notations and preliminaries are listed in their respective background sections.

2.1 Notations

We consider a single-relation schema $\mathcal{A} = (A_1, \ldots, A_m)$, which is a vector of distinct attribute names A_i, each associated with a domain $\texttt{dom}(A_i)$ of values. A database D over $\mathcal{A}$ is a associated with a set $tids(D)$ of *tuple identifiers*, and it maps every identifier $i \in tids(D)$ to a tuple $D[i] = (a_1, \ldots, a_m)$ in $A_1 \times \cdots \times A_m$. A database D' is a *subset* of D, denoted $D' \subseteq D$, if D' is obtained from D by deleting zero or more tuples, that is, $tids(D') \subseteq tids(D)$ and $D'[i] = D[i]$ for all $i \in tids(D')$. We also use $S_{:i}$ to denote all elements from 1 to i in a sequence S. In Table 2.1, we list the common notations in the thesis.

2.2 Differential Privacy

Differential privacy (DP) was first introduced in 2006[52] as a promise to ensure the private information of an individual while allowing the learning of useful information about a population. This promise has quickly arisen as the state of the art privacy definition with

Table 2.1: Table of notations

$\mathcal{D}$	The set of possible dataset instances
D	A database instance
$\mathcal{A}$	The set of attributes in a dataset
n	Number of rows in dataset
$\texttt{dom}(A_i)$	Domain of attribute A_i
$\mathcal{M}$	A randomized privacy mechanism often used as part of a privacy definition
ε, δ	Differential privacy parameters also called privacy budget
Δ_f	Sensitivity of a function f according to Definition 2
L	Lot size or batch size of an ML optimizer
σ	Noise parameter/scale used to add privacy noise in DP optimizer
C	Clipping norm as part of DP optimizers

a rich class of mechanisms satisfying it. Unlike anonymization, this privacy guarantee specifies a provable property of the privacy-preserving mechanisms.

2.2.1 Definition

An algorithm satisfies differential privacy if adding, removing or changing a record in terms of the input does not significantly alter the output of the algorithm. More formally, we define DP in terms of two neighboring datasets, D and D' (denoted by $D \approx D'$) if D' can be transformed from D by adding or removing one tuple in D

Definition 1 (Differential Privacy). *An algorithm $\mathcal{M}$ is said to satisfy ε-DP if for all $S \subseteq Range(\mathcal{M})$ and for all $D \approx D'$,*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta\,.$$

In this definition, the number of individuals in the database are unknown, and hence this definition is also known as *unbounded differential privacy*. When the number of the individuals is known in the database, the neighboring databases are defined as a pair of databases that differ in the value of only one individuals record, and the remaining individuals all have the same record values. This is also known as *bounded differential privacy*. Although, these definitions do not affect the privacy guarantee much but are important to keep in mind while implementation of the algorithms. We will be using the notion of unbounded-DP for Chapter 3 and bounded-DP for the rest.

The privacy cost is measured by the parameters (ε, δ), often referred to also as the privacy budget. The smaller the privacy parameters, the stronger the offered privacy. When $\delta = 0$, we call the privacy measure pure-DP otherwise approx-DP.

2.2.2 Algorithms for DP

A common way to achieve DP is by adding noise to the output a mechanism. This requires to measure the change in a particular function's result over two neighboring databases. The supremum over all pairs of neighboring databases is called the *sensitivity* of the function.

Definition 2 (Global sensitivity [54]). *Given a function $f : \mathcal{D} \to \mathbb{R}$, the sensitivity of f is*

$$\Delta_f = \max_{D' \approx D} \|f(D) - f(D')\|_p. \tag{2.1}$$

Laplace mechanism [53] and Gaussian mechanism [55] are two widely used noise addition DP algorithms. Laplace mechanism uses the ℓ_1 ($p = 1$ in Definition 2) sensitivity and achieves pure-DP whereas the Gaussian mechanism uses the ℓ_2 ($p = 2$ in Definition 2) sensitivity and archives approx-DP.

Laplace mechanism. The Laplace mechanism [54] is a common building block in DP mechanisms and is used to get a noisy estimate for queries with numeric answers. The noise injected is calibrated to the query's global sensitivity.

Definition 3 (Laplace Mechanism [54]). *Given a database D, a function $f : \mathcal{D} \to \mathbb{R}$, and a privacy budget ε, the Laplace mechanism $\mathcal{M}_L$ returns $f(D) + \nu_q$, where $\nu_q \sim Lap(\Delta_f/\varepsilon)$ and Δ_f is the sensitivity computed using ℓ_1-norm.*

The Laplace mechanism can answer many numerical queries, but the exponential mechanism can be used in many natural situations requiring a non-numerical output.

Gaussian Mechanism. The Gaussian Mechanism is similar to Laplace but uses the $\ell - 2$ sensitivity and achieves approx-DP.

Definition 4 (Gaussian Mechanism [55]). *Given a database D, a function $f : \mathcal{D} \to \mathbb{R}$, and privacy budget ε, δ, the Gaussian mechanism $\mathcal{M}_G$ returns outputs $f(D) + b$, where $b \sim \mathcal{N}(0, \sigma^2 \Delta_f)$, Δ_f is the sensitivity computed using ℓ_2-norm and σ is the noise scale.*

For $\varepsilon \in (0, 1)$, the $\sigma \geq \sqrt{2 \ln \frac{1.25}{\delta}}/\varepsilon$, the Gaussian mechanism satisfies (ε, δ)-DP.

In many natural situations, however, we would like a non-numerical output such as a probabilistic selection of an object from a set of objects evaluated over a private database. The exponential mechanism helps in such situations.

Exponential mechanism. The exponential mechanism [121] expands the application of DP by allowing a non-numerical output.

Definition 5 (Exponential Mechanism [121]). *Given a dataset D, a privacy budget ε, a set Θ of output candidates, a quality function $q(D, \theta_i) \in \mathbb{R}$, the exponential mechanism $\mathcal{M}_{EM}$ outputs a candidate $\theta_i \in \Theta$ with probability proportional to $\exp\left(\frac{\varepsilon q(D,\theta_i)}{2\Delta_q}\right)$, where Δ_q is the sensitivity of the quality function q.*

The exponential mechanism offers strong utility guarantees proportional to the number of candidates $|\Theta|$ and the sensitivity of the quality function Δq as formalized by Theorem 1.

Theorem 1 ([121]). *Let D be a private dataset, and $OPT(D) = \max_{\theta\in\Theta} q(D, \theta)$ be the score attained by the best object θ with respect to the dataset D due to the exponential mechanism $M_{EM}(D)$. If the set of objects that achieve the $OPT(D)$, $\Theta^* = \{\theta \in \Theta : q(D, \theta) = OPT(D)\}$ has size $|\Theta^*| \geq 1$. Then*

$$\Pr[q(D, M_E(D)) \leq OPT(D) - \frac{2\Delta_q}{\varepsilon_{em}}(\ln|\Theta| + t)] \leq \exp(-t)$$

where ε_{em} is the privacy budget for the exponential mechanism and Δ_q is the sensitivity of the quality function q.

2.2.3 Properties of DP

Complex DP algorithms can be built from the basic algorithms following two essential properties of differential privacy:

Proposition 1 (DP Properties [50, 51]). *The following hold.*

1. **(Sequential composition)** *If $\mathcal{M}_i$ satisfies ε_i-DP, then the sequential application of $\mathcal{M}_1$, $\mathcal{M}_2, \cdots$, satisfies $(\sum_i \varepsilon_i)$-DP.*
2. **(Parallel composition)** *If each $\mathcal{M}_i$ accesses disjoint sets of tuples, they satisfy $(\max_i \varepsilon_i)$-DP together.*
3. **(Post-processing)** *Any function applied to the output of an ε-DP mechanism $\mathcal{M}$ also satisfies ε-DP.*

2.2.4 DP for Graphs

When the dataset is a graph $\mathcal{G} = (V, E)$, the standard definition can be translated to two variants of DP [75]. The first is *edge-DP* where two graphs are neighboring if they differ on one edge, and the second is *node-DP*, when two graphs are neighboring if one is obtained from the other by removing a node (and its incident edges). The two definitions offer different kinds of privacy protection. In our work for computing inconsistency measures in Chapter 3, we deal with databases and their corresponding conflict graphs, adding or removing a tuple of the dataset translates to node-DP. The corresponding definition of neighboring datasets changes to neighboring graphs, where two graphs $\mathcal{G}$ and $\mathcal{G}'$ are called neighboring $\mathcal{G} \approx \mathcal{G}'$ if $\mathcal{G}'$ can be transformed from $\mathcal{G}$ by adding or removing one node along with all its edges in $\mathcal{G}$. Node-DP provides a stronger privacy guarantee than edge-DP since it protects an individual's privacy and all their connections, whereas edge-DP concerns only one such connection.

Definition 6 (Node sensitivity)**.** *Given a function f over a graph $\mathcal{G}$, the sensitivity of f is* $\Delta_f = \max_{\mathcal{G}' \approx \mathcal{G}} |f(\mathcal{G}) - f(\mathcal{G}')|$.

The building blocks of DP, such as the Laplace and Exponential mechanisms, also work on graphs by simply substituting the input to a graph and the sensitivity to the corresponding node sensitivity.

Graph projection. Graph projection algorithms refer to a family of algorithms that help reduce the node sensitivity of a graph by truncating the edges and, hence, bounding the maximum degree of the graph. Several graph projection algorithms exist [92, 25], among which the *edge addition* algorithm [42] stands out for its effectiveness in preserving most of the underlying graph structure. The edge addition algorithm denoted by π_θ^Λ, takes as input the graph $\mathcal{G} = \mathcal{G}_\Sigma^D = (V, E)$, a bound on the maximum degree of each vertex (θ), and a stable ordering of the edges (Λ) to output a projected θ-bounded graph denoted by $\mathcal{G}_\theta = \pi_\theta^\Lambda(\mathcal{G})$.

Definition 7 (Stable ordering [42])**.** *A graph edge ordering Λ is stable if and only if given two neighboring graphs $\mathcal{G} = (V, E)$ and $\mathcal{G}' = (V', E')$ that differ by only a node, $\Lambda(\mathcal{G})$ and $\Lambda(\mathcal{G}')$ are consistent in the sense that if two edges appear both in $\mathcal{G}$ and $\mathcal{G}'$, their relative ordering are the same in $\Lambda(\mathcal{G})$ and $\Lambda(\mathcal{G}')$.*

The stable ordering of edges, $\Lambda(\mathcal{G})$, can be any deterministic ordering of all the edges E in the $\mathcal{G}$. Such stabling edge ordering can be easily obtained in practice. For example, it could be an ordering (e.g. alphabetical ordering) based on the node IDs of the graph such

that in the neighboring dataset $\mathcal{G}'$, the edges occur in the same ordering as $\mathcal{G}$. The edge addition algorithm starts with an empty set of edges and operates by adding edges in the same order as Λ so that each node has a maximum degree of θ. To simplify the notation, in the remainder of the paper, we drop Λ and denote the edge addition algorithm $\pi_\theta^\Lambda(\mathcal{G})$ as $\pi_\theta(\mathcal{G})$.

Algorithm 1: Edge addition algorithm [42]

Data: Graph $\mathcal{G}(V, E)$, Bound θ, Stable ordering Λ
Result: θ-bounded graph $\pi_\theta(\mathcal{G})$

```
E^θ ← ∅; d(v) ← 0 for each v ∈ V ;
for e = (u, v) ∈ Λ do
    if d(u) < θ & d(v) < θ then
        E^θ ← E^θ ∪ {e} d(u) ← d(u) + 1, d(v) ← d(v) + 1 ;
return G^θ = (V, E^θ);
```

2.2.5 DP for Machine Learning

In this thesis, machine learning predominantly refers to training neural networks. Neural networks are essentially a set of parameters that are iteratively updated such that the input, when multiplied by these parameters, maps to the output. Depending on the task at hand (classification, regression, generation, etc.), the input and output may differ. The parameters are updated using the standard gradient descent procedure using a gradient optimizer. In each iteration, a batch of input points is passed to the network to get a predicted output, and a loss value is calculated based on the difference between the predicted and the true output. The optimizer then calculates the gradient of the loss function with respect to the network parameters and takes a step in the negative direction of the gradient to update the network parameters.

In contrast to non-private optimizers, where batches are sliced from the training dataset, DP optimizers at each iteration work by sampling "lots" from the training with probability L/n, where L is the (expected) lot size and n is the total data size. The gradients are then computed similarly to the non-private counterpart based on the loss function and task. As there is no a priori bound on these query outputs, the sensitivity Δ_f is set by clipping the maximum ℓ_2 norm of the gradient to a user-defined parameter C. The gradient of each point is then noised and published. All DP optimizers follow the same framework in which they take steps on the computed noisy gradient as in its non-private counterpart [119].

The privacy cost of the whole training procedure is calculated by advanced composition techniques such as the Moments accountant [5].

Moments Accountant [5]. The moments accountant based on Rényi-DP [122] keeps track of a bound on the moments of the privacy loss random variable (In DP machine learning, gradients). Using the strong composition theorem, it generalizes the standard approach of tracking (ε, δ). Specifically, the moments accountant computes the tail bound of the privacy loss variable by composing the log moments. These log moments compose linearly and provide a much stricter bound on the privacy loss variable than the loose naive composition. While such an improvement was known previously for composing Gaussian mechanisms, it also applies to composing Gaussian mechanisms with random sampling. It can provide a much tighter estimate of the privacy loss in iterative processes like gradient descent. In recent work, other accountants have also been proposed for tighter and average-case compositions [12, 69]. In chapter 5, we will study in detail how moments accountant compares to other techniques for hyperparameter tuning.

2.2.6 DP for Synthetic Data Generation

Another standard DP learning paradigm is generating synthetic data given a fixed privacy budget. The synthetic data, once generated, can be made public, and all queries on this dataset come for free due to the post-processing property of DP. There are three main approaches for DP synthetic data generation [168]:

Statistical approaches rely on estimating low-dimensional statistics about the dataset such as marginals [185, 150, 117]. These approaches can be made better by finding the correlation between attributes. Techniques for improvement include probabilistic models [95], Bayesian models [193, 103, 147] and undirected graphs [36, 118]. Statistical approaches capture the underlying distribution of the correlated independent attributes very well but fail to imbibe complex relationships between multiple attributes.

Deep learning approaches are promising for generating synthetic data [160, 33, 73], particularly with autoencoders and generative adversarial networks (GAN). Autoencoders map data into a low-dimensional feature space and sample synthetic data from the low-dimensional space. GANs utilize a generator to produce fake examples and a discriminator to distinguish real from fake. DPSGD [181, 20, 162, 6] is commonly used for privacy. Various private approaches for autoencoders [146, 7, 9] and GANs [61, 87, 173, 186] have been proposed, effective on image data but challenged with tabular data due to poor encoding. Conditional GANs [187] and private versions [172] address this by sampling based on conditional probabilities of categorical attributes.

Mixed approaches are inspired by both the above approaches and try to preserve both low-dimensional statistics and high-level information. Some techniques include leveraging the dimensionality reduction via random orthonormal (RON) projection, the Gaussian generative model [30], combining denial constraints and attribute-wise embedding models [62] and Gretel.ai statistics [134].

Chapter 3

Data Exploration and Cleaning: Computing Inconsistency Measures

3.1 Introduction

The utility of sensitive data primarily depends on its quality. Organizations that build applications on spend vast amounts of money on purchasing data from data marketplaces [108, 164, 171, 184]. These marketplaces build relationships and manage monetary transactions between data owners and buyers. These buyers are often organizations that want to develop applications such as machine learning models or personalized assistants. Before the buyer purchases a dataset at a specific cost, they may want to ensure the data is suitable for their use case, adhere to particular data quality constraints, and be able to profile its quality to know if the cost reflects the quality.

To address such scenarios, we consider the problem of assessing the quality of databases protected by DP. Such quality assessment will allow users to decide whether they can rely on the conclusions drawn from the data or whether the suggested data is suitable for them. To solve this problem, we must tackle several challenges. First, since DP protects the database, users can only observe noisy aggregate statistics, which can be challenging to summarize into a quality score. Second, if the number of constraints is large (e.g., if they were generated with an automatic system [24, 111, 145]), translating each constraint to an SQL `COUNT` query and evaluating it over the database with a DP mechanism may

lead to low utility since the number of queries is large, allowing for only a tiny portion of the privacy budget to be allocated to each query.

Hence, our proposed solution employs *inconsistency measures* [170, 142, 114, 22, 70, 71, 112, 110] that quantify data quality with a single number for all constraints, essentially yielding *a data quality score.* This approach aligns well with DP, as such measures give a single aggregated numerical value representing data quality, regardless of the given number of constraints. As inconsistency measures, we adopt the ones studied by Livshits et al˙ [114] following earlier work on the topic [170, 142, 22, 70]. This work discusses and studies five measures, including (1) the *drastic measure*, a binary indicator for whether the database contains constraint violations, the (2) *maximal consistency measure*, counting the number of maximal tuple sets for which addition of a single tuple will cause a violation, the (3) *minimal inconsistency measure*, counting the number of minimal tuple sets that violate a constraint, the (4) *problematic measure*, counting the number of constraint violations, the (5) *minimal repair measure*, counting the minimal tuple deletions needed to achieve consistency. These measures apply to various inconsistency measures that have been studied in the literature of data quality management, including functional dependencies, the more general conditional functional dependencies [27], and the more general denial constraints [37]. We show that the first two measures are incompatible for computation in the DP setting (Section 3.3), focusing throughout the paper on the latter three.

An approach that one may suggest to computing the inconsistency measures in a DP manner is to translate the measure into an SQL query and then compute the query using an SQL engine that respects DP [167, 47, 97, 85]. Specifically relevant is R2T [47], the state-of-the-art DP mechanism for such queries that supports select-project-join-aggregate (SPJA) queries, including self-joins. Nevertheless, when considering the three measures of inconsistency we focus on, this approach has several drawbacks. One of these measures (number of problematic tuples) requires the SQL `DISTINCT` operator that R2T cannot handle. In contrast, another measure (minimal repair) cannot be expressed at all in SQL, making such engines irrelevant.

Contrasting the SQL approach, the approach we propose and investigate here models the violations of the integrity constraints as a *conflict graph* and applies DP techniques for graph statistics. In the conflict graph, nodes are tuple identifiers, and there is an edge between a pair of tuples if this pair violates a constraint. Then, each inconsistency measure can be mapped to a specific graph statistic. Using this view of the problem allows us to leverage prior work on releasing graph statistics with DP [75, 92, 42] and develop tailored mechanisms for computing inconsistency measures with DP.

To this end, we harness graph projection techniques from the state-of-the-art DP al-

Table 3.1: Relative errors for a SQL approach vs our approach to compute the minimal inconsistency measure at $\varepsilon = 1$

DP Algorithms	**Adult [21]**	**Flight [136]**	**Stock [137]**
R2T [47]	0.17 ± 0.01	0.12 ± 0.03	123.19 ± 276.73
This work	0.10 ± 0.05	0.10 ± 0.20	0.07 ± 0.08

gorithms [42] that truncate the graph to achieve DP. While these algorithms have proven effective in prior studies on social network graphs, they may encounter challenges with conflict graphs arising from their unique properties. To overcome this, we devise a novel optimization for choosing the truncation threshold. We further provide a DP mechanism for the minimal repair measure that augments the classic 2-approximation of the vertex cover algorithm [179] to restrict its sensitivity and allow effective DP guarantees with high utility. Our experimental study shows that our novel algorithms prove efficacious for different datasets with various conflict graph sizes and sparsity levels.

Beyond handling the two inconsistency measures that R2T cannot handle, our approach provides considerable advantages even for the one R2T can handle (number of conflicts). For illustration, Table 3.1 shows the results of evaluating R2T [47] on three datasets with the same privacy budget of 1 for this measure. Though R2T performed well for the Adult and Flight datasets, it reports more than 120% relative errors for the Stock dataset with very few violations. On the other hand, our approach demonstrates strong performance across all three datasets.

3.2 Preliminaries

We begin with some background that we need to describe the concept of inconsistency measures for private databases.

3.2.1 Database Constraints

Following previous work on related topics [63, 111], we focus on Denial Constraints (DCs) on pairs of tuples. Using the formalism of Tuple Relational Calculus, such a DC is of the form $\forall t, t' \neg\big(\varphi_1 \wedge \cdots \wedge \varphi_k\big)$ where each term φ_j is a comparison $\nu_1 \circ \nu_2$ so that: (a) each of ν_1 and ν_2 is either $t[A_i]$, or $t'[A_i]$, or a, where A_i is some attribute, $t[A_i]$ represents the value of A_i attribute in tuple t and a is a constant value, and (b) the operator $\circ$ belongs to set $\{<, >, \leq, \geq, =, \neq\}$ of comparisons. This DC states that there cannot be two tuples t and t' such that all comparisons φ_j hold true (i.e., at least one φ_j should be violated).

ID	Capital	Country
1	Ottawa	Canada
2	Ottawa	Canada
3	Ottawa	Canada
4	Ottawa	Kanada

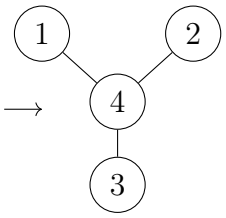


Figure 3.1: Toy example dataset to show a worst-case analysis. An additional row may violate all other rows in the dataset (left). Easier analysis can be done by instead converting the dataset into its corresponding conflict graph (right).

Note that the class of DCs of the form that we consider generalizes the class of Functional Dependencies (FDs). An FD has the form $X \to Y$ where $X, Y \subseteq \{A_1, \ldots, A_m\}$, and it states that every two tuples that agree on (i.e., have the same value in each attribute of) X must also agree on Y.

In the remainder of the chapter, we denote by Σ the given set of DCs. A database D *satisfies* Σ, denoted $D \models \Sigma$, if D satisfies every DC in Σ; otherwise, D *violates* Σ, denoted $D \not\models \Sigma$.

A common way of capturing the violations of Σ in D is through the *conflict graph* $\mathcal{G}_\Sigma^D$, which is the graph (V, E), where $V = tids(D)$ an edge $e = \{i, j\} \in E$ occurs whenever the tuples $D[i]$ and jointly $D[j]$ violate Σ. To simplify the notation, we may write simply $\mathcal{G}$ instead of $\mathcal{G}_\Sigma^D$ when there is no risk of ambiguity.

Example 2. *Consider a dataset that stores information about capital and country as shown in Figure 3.1. Assume an FD constraint σ : Capital $\to$ Country between attributes capital and country that says that the country of two tuples must be the same if their capital is the same. Assume the dataset has 3 rows (white color) and a neighboring dataset has an extra row (grey color) with the typo in its country attribute. As shown in the right side of Figure 3.1, the dataset with 4 rows can be converted to a conflict graph with the nodes corresponding to each tuple and edges referring to conflicts between them. The $\mathcal{I}_{\mathsf{MI}}$ measure computes the size of the set of all minimally inconsistent subsets (the number of edges in the graph) for this dataset.*

3.2.2 Inconsistency Measures

Inconsistency measures have been studied in previous work [22, 70, 71, 112, 110] as a means of measuring database quality for a set of DCs. We adopt the measures and notation of Livshits et al. [114]. Specifically, they consider five inconsistency measures that capture

different aspects of the dataset quality. To define these concepts, we need some notation. Given a database D and a set Σ of anti-monotonic integrity constraints, we denote by $\mathsf{MI}_\Sigma(D)$ the set of all *minimally inconsistent subsets*, that is, the sets $E \subseteq D$ such that $E \not\models \Sigma$ but $E' \models \Sigma$ for all $E' \subsetneq E$. We also denote by $\mathsf{MC}_\Sigma(D)$ the set of all *maximal consistent subsets* of D; that is, the sets $E \subseteq D$ such that $E \models \Sigma$ and $E' \not\models \Sigma$ whenever $E \subsetneq E' \subseteq D$.

Definition 8 (Inconsistency measures [114])**.** *Given a database D and a set of DCs Σ, the inconsistency measures are defined as follows:*

- *Drastic measure:* $\mathcal{I}_\mathsf{D}(D, \Sigma) = 1$ *if* $D \models \Sigma$ *and 0 otherwise.*
- *Minimal inconsistency measure:* $\mathcal{I}_\mathsf{MI}(D, \Sigma) = |\mathsf{MI}_\Sigma(D)|$.
- *Problematic measure:* $\mathcal{I}_\mathsf{P}(D, \Sigma) = |\cup \mathsf{MI}_\Sigma(D)|$.
- *Maximal consistency measure:* $\mathcal{I}_\mathsf{MC}(D, \Sigma) = |\mathsf{MC}_\Sigma(D)|$ [1].
- *Optimal repair measure:* $\mathcal{I}_\mathsf{R}(D, \Sigma) = |D| - |D_R|$, *where* $|D_R|$ *is the largest subset* $D_R \subseteq D$ *such that* $D_R \models \Sigma$.

Observe that inconsistency measures also have a graphical interpretation for the conflict graph $\mathcal{G}_\Sigma^D$. For instance, the drastic measure $\mathcal{I}_\mathsf{D}(D, \Sigma)$ corresponds to a binary indicator for whether there exists an edge in $\mathcal{G}_\Sigma^D$. We summarize the graph interpretation of these inconsistency measures in Table 3.2.

Table 3.2: Summary of Inconsistency Measures, $n = |D| = |\mathcal{G}_\Sigma^D.V|$, $m = |\mathcal{G}_\Sigma^D.E|$, Θ is the candidate set.

	Non-private analysis [114]		**DP analysis (this work)**		
Measure	**Graph Interpretation**	**Computation**	**Sensitivity**	**Computation**	**Utility**
$\mathcal{I}_\mathsf{D}$	If exists an edge	$O(\|\Sigma\|n^2)$	1	N.A.	N.A.
$\mathcal{I}_\mathsf{MI}$	No. of edges	$O(\|\Sigma\|n^2)$	n	$O(\|\Sigma\|n^2 + \|\Theta\|m)$	$-\tilde{q}_{\text{opt}}(D, \varepsilon_2)$ $+O(\frac{\theta_{\max} \ln \|\Theta\|}{\varepsilon_1})$
$\mathcal{I}_\mathsf{P}$	No. of nodes with positive degrees	$O(\|\Sigma\|n^2)$	n	$O(\|\Sigma\|n^2 + \|\Theta\|m)$	$-\tilde{q}_{\text{opt}}(D, \varepsilon_2)$ $+O(\frac{\theta_{\max} \ln \|\Theta\|}{\varepsilon_1})$
$\mathcal{I}_\mathsf{MC}$	No. of maximal independent sets	#P-complete	$O(3^n)$	N.A.	N.A.
$\mathcal{I}_\mathsf{R}$	Minimum vertex cover size	NP-hard	1	$O(\|\Sigma\|n^2 + m)$	$\mathcal{I}_R(D, \Sigma)$ $+O(1/\varepsilon)$

[1] We drop "-1" from the original definition [114] for simplicity.

3.3 Inconsistency Measures under DP

Problem Setup. Consider a private dataset D, a set of DCs Σ, and a privacy budget ε. For an inconsistency measure $\mathcal{I}$ from the set $\{\mathcal{I}_{\mathsf{D}}, \mathcal{I}_{\mathsf{MI}}, \mathcal{I}_{\mathsf{P}}, \mathcal{I}_{\mathsf{MC}}, \mathcal{I}_{\mathsf{R}}\}$ (Definition 8), we would like to design an ε-DP algorithm $\mathcal{M}(D, \Sigma, \varepsilon)$ such that with high probability, $|\mathcal{M}(D, \Sigma, \varepsilon) - \mathcal{I}(D, \Sigma)|$ is bounded with a small error.

Sensitivity Analysis. We first analyze the sensitivity of the five inconsistency measures and discuss the challenges to achieving DP.

Proposition 2. *Given a database D and a set of DCs Σ, where $|D| = n$, the following holds:*

1. *The global sensitivity of $\mathcal{I}_{\mathsf{D}}$ is 1.*
2. *The global sensitivity of $\mathcal{I}_{\mathsf{MI}}$ is n.*
3. *The global sensitivity of $\mathcal{I}_{\mathsf{P}}$ is n.*
4. *The global sensitivity of $\mathcal{I}_{\mathsf{MC}}$ is exponential in n.*
5. *The global sensitivity of $\mathcal{I}_{\mathsf{R}}$ is 1.*

Proof. Consider two neighbouring datasets, D and D'. The proof for each measure is shown below in separate paragraphs:

For $\mathcal{I}_{\mathsf{D}}$: Adding or removing one tuple from the dataset will affect the addition or removal of all conflicts related to it in the dataset. In the worst case, this tuple could remove all conflicts in the dataset, and the $\mathcal{I}_{\mathsf{D}}$ could go from 1 to 0 or vice versa.

For $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$: The $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ are concerned with the set of minimally inconsistent subsets $MI_{\Sigma}(D)$. The $\mathcal{I}_{\mathsf{MI}}$ measure computes the total number of inconsistent subsets $|MI_{\Sigma}(D)|$ and $\mathcal{I}_{\mathsf{P}}$ computes the total number of unique rows participating in $MI_{\Sigma}(D)$, $|\cup MI_{\Sigma}(D)|$. Now, without loss in generality, let's assume D' has an additional tuple compared to D. In the worst case, the extra row could violate all other rows in the dataset, adding $|D|$ inconsistent subsets to $MI_{\Sigma}(D)$. Therefore, in the worst case, the $\mathcal{I}_{\mathsf{MI}}$ measure and $\mathcal{I}_{\mathsf{P}}$ could change by $|D|$.

For $\mathcal{I}_{\mathsf{MC}}$: The $\mathcal{I}_{\mathsf{MC}}$ measure is #P-complete and can be computed for a conflict graph $\mathcal{G}_{\Sigma}^{D}$ if the dataset has only FDs in the constraint set Σ and $\mathcal{G}_{\Sigma}^{D}$ is P_4-free [93]. The maximum number of maximal independent sets [128, 72] $f(n)$ for a graph with n vertices is given by:

$$\text{If } n \geqq 2, \text{ then } f(n) = \begin{cases} 3^{n/3}, & \text{if } n \equiv 0(\text{mod}3); \\ 4.3^{[n/3]-1}, & \text{if } n \equiv 1(\text{mod}3); \\ 2.3^{[n/3]}, & \text{if } n \equiv 2(\text{mod}3). \end{cases}$$

Using the above result, we can see that adding or removing a node in the graph can affect the total number of maximal independent sets in the order of 3^n.

For $\mathcal{I}_{\mathsf{R}}$: Without loss in generality, let's assume D' has an additional tuple compared to D. This extra row may or may not add extra violations. However, repairing this extra row in D' will always remove these added violations. Therefore, $\mathcal{I}_{\mathsf{R}}$ for D' can only be one extra than D. □

Inadequacy of $\mathcal{I}_{\mathsf{D}}$ and $\mathcal{I}_{\mathsf{MC}}$. We note that two inconsistency measures are less suitable for DP. First, the drastic measure $\mathcal{I}_{\mathsf{D}}$ is a binary measure that outputs 1 if at least one conflict exists in the dataset and 0 otherwise. Due to its binary nature, the measure's sensitivity is 1, meaning adding or removing a single row can significantly alter the result. Adding DP noise to such a binary measure can render it meaningless.

One way to compute the $\mathcal{I}_{\mathsf{D}}$ measure could be to consider a proxy of $\mathcal{I}_{\mathsf{D}}$ by employing a threshold-based approach that relies on $\mathcal{I}_{\mathsf{P}}$ or $\mathcal{I}_{\mathsf{MI}}$. For example, if these measures are below a certain given number, we return 0 and, otherwise, 1. A recent work [144] addresses similar problems for synthetic data generation by employing the exponential mechanism. However, since we focus on directly computable measures in the DP setting, we leave this intriguing subject for future work.

Additionally, prior work [110] showed that computing $\mathcal{I}_{\mathsf{MC}}$ is #P-complete, restricting its practicality. Even in special cases where $\mathcal{I}_{\mathsf{MC}}$ can be polynomially computed (when $\mathcal{G}_{\Sigma}^{D}$ is P_4-free [93]), its global sensitivity is exponential in the number of nodes of $\mathcal{G}_{\Sigma}^{D}$, diminishing the utility of its DP estimate. Therefore, we defer the study of $\mathcal{I}_{\mathsf{D}}$ and $\mathcal{I}_{\mathsf{MC}}$ to future work.

Challenges for $\mathcal{I}_{\mathsf{R}}$, $\mathcal{I}_{\mathsf{MI}}$, and $\mathcal{I}_{\mathsf{P}}$. Although the $\mathcal{I}_{\mathsf{R}}$ measure has a low sensitivity of 1 for its output range $[0, n]$, it is an NP-hard problem, and the common non-private solution is to solve a linear approximation that requires solving a linear program [110]. However, in the worst case, this linear program again has sensitivity equal to n (number of rows in the dataset) and may have up to $\binom{n}{2}$ number of constraints (all rows violating each other).

Existing state-of-the-art DP linear solvers [78] are slow and fail for such a challenging task. Our preliminary experiments to solve such a linear program timed out after 24 hours with $n = 1000$. For $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ they have polynomial computation costs and reasonable output ranges. However, they still have high sensitivity n. In the upcoming sections 3.4 and 3.5, we show that these problems can be alleviated by pre-processing the input dataset as a conflict graph and computing these inconsistency measures as private graph statistics.

3.4 DP graph projection for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$

Computing graph statistics such as edge count and degree distribution while preserving node-differential privacy (node-DP) is a well-explored area [42, 92, 25]. Hence, in this section, we leverage the state-of-the-art node-DP approach for graph statistics to analyze the inconsistency measures $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ represented as graph statistics on the conflict graph $\mathcal{G}_\Sigma^D$. However, the effectiveness of this approach hinges on carefully chosen parameters. We introduce two optimization techniques that consider the constraints imposed on the database D to optimize parameter selection and enhance the algorithm's utility.

3.4.1 Graph Projection Approach for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$

A primary utility challenge in achieving node-DP for graph statistics is their high sensitivity. In the worst case, removing a single node from a graph of n nodes can result in removing $(n-1)$ edges. To mitigate this issue, the state-of-the-art approach [42] first projects the graph $\mathcal{G}$ onto a θ-bounded graph $\mathcal{G}_\theta$, where the maximum degree is no more than θ. Subsequently, the edge count of the transformed graph is perturbed by the Laplace mechanism with a sensitivity value of less than n. However, the choice of θ is critical for accurate estimation. A small θ reduces Laplace noise due to lower sensitivity, but results in significant edge loss during projection. Conversely, a θ close to n preserves more edges but increases the Laplace noise. Prior work addresses this balance using the exponential mechanism (EM) to prefer a θ that minimizes the combined errors arising from graph projection and the Laplace noise.

We outline this general approach in Algorithm 2. This algorithm takes in the dataset D, the constraint set Σ, a candidate set Θ for degree bounds, and privacy budgets ε_1 and ε_2. These privacy budgets are later composed to get a final guarantee of ε-DP. We start by constructing the conflict graph $\mathcal{G}_\Sigma^D$ generated from the input dataset D and constraint set Σ (line 1), as defined in Section 3.2.1. Next, we sample in a DP manner a value of

Algorithm 2: Graph projection approach for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$

Data: Dataset D, constraint set Σ, candidate set Θ, privacy budgets ε_1 and ε_2
Result: DP inconsistency measure for $\mathcal{I}_{\mathsf{MI}}$ or $\mathcal{I}_{\mathsf{P}}$

1 Construct the conflict graph $\mathcal{G}_\Sigma^D$
2 Sample θ^* from Θ with a ε_1-DP mechanism // Basic EM (Algorithm 3); Optimized EM (Algorithm 4)
3 Compute θ^*-bounded graph $\mathcal{G}_{\theta^*} \leftarrow \pi_{\theta^*}(\mathcal{G}_\Sigma^D)$ // Edge addition algorithm [42]
4 **Return** $f(\mathcal{G}_{\theta^*}) + \mathrm{Lap}(\frac{\theta^*}{\varepsilon_2})$ // $f(\cdot)$ returns edge count for $\mathcal{I}_{\mathsf{MI}}$ and the number of nodes with positive degrees for $\mathcal{I}_{\mathsf{P}}$

Algorithm 3: EM-based first try for parameter selection

Data: Graph $\mathcal{G}$, candidate set Θ, quality function q, privacy budget $\varepsilon_1, \varepsilon_2$
Result: Candidate θ^*

1 Find the maximum value in Θ as $\theta_{\max}$
2 For each $\theta_i \in \Theta$, compute $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ // See Equation (3.1)
3 Sample θ^* with prob $\propto \exp(\frac{\varepsilon_1 q_{\varepsilon_2}(\mathcal{G}, \theta_i)}{2\theta_{\max}})$
4 **Return** θ^*

θ^* from the candidate set Θ with the privacy budget ε_1 (line 2). A baseline choice is an exponential mechanism detailed in Algorithm 3 to output a degree that minimizes the edge loss in a graph and the Laplace noise. In line 3, we compute a bounded graph $\mathcal{G}_{\theta^*}$ using the edge addition algorithm [42], we compute a θ^*-bounded graph $\mathcal{G}_{\theta^*}$ (detailed in Section 3.2). Finally, we perturb the true measure (either the number of edges for $\mathcal{I}_{\mathsf{MI}}$ or the number of positive degree nodes for $\mathcal{I}_{\mathsf{P}}$) on the projected graph, denoted by $f(\mathcal{G}_{\theta^*})$, by adding Laplace noise using the other privacy budget ε_2 (line 4).

The returned noisy measure at the last step has two sources of errors: (i) the bias incurred in the projected graph, i.e., $f(\mathcal{G}) - f(\mathcal{G}_{\theta^*})$, and (ii) the noise from the Laplace mechanism with an expected square root error $\sqrt{2}\theta^*/\varepsilon_2$. Both errors depend on the selected parameter θ^*, and it is vital to select an optimal θ^* that minimizes the combined errors. Next, we describe a DP mechanism that helps select this parameter.

EM-based first try for parameter selection. The EM (Definition 5) specifies a quality function $q(\cdot, \cdot)$ that maps a pair of a database D and a candidate degree θ to a numerical value. The optimal θ value for a given database D should have the largest possible quality value and, hence, the highest probability of being sampled. We also denote $\theta_{\max}$ the largest degree candidate in Θ and use it as part of the quality function to limit its sensitivity.

The quality function we choose to compute the inconsistency measures includes two

θ	e_{bias}	$\sqrt{2}\theta/\varepsilon_2$	q
1	2	$\sqrt{2}$	$-2-\sqrt{2}$
2	1	$2\sqrt{2}$	$-1-2\sqrt{2}$
3	0	$3\sqrt{2}$	$-3\sqrt{2}$

Table 3.3: Quality function computation for $\mathcal{I}_{\mathsf{MI}}$ for the conflict graph in Figure 3.1 when $\varepsilon_2 = 1$

terms: for each $\theta \in \Theta$,

$$q_{\varepsilon_2}(\mathcal{G}, \theta) = -e_{\text{bias}}(\mathcal{G}, \theta) - \sqrt{2}\theta/\varepsilon_2 \tag{3.1}$$

where the first term e_{bias} captures the bias in the projected graph, and the second term $\sqrt{2}\theta/\varepsilon_2$ captures the error from the Laplace noise at budget ε_2. For the minimum inconsistency measure $\mathcal{I}_{\mathsf{MI}}$, we define the bias term as

$$e_{\text{bias}}(\mathcal{G}, \theta) = |\mathcal{G}_{\theta_{\max}}.E| - |\mathcal{G}_{\theta}.E| \tag{3.2}$$

i.e., the number of edges truncated at degree θ as compared to that at degree $\theta_{\max}$. For the problematic measure $\mathcal{I}_{\mathsf{P}}$, we have

$$e_{\text{bias}}(\mathcal{G}, \theta) = |\mathcal{G}_{\theta_{\max}}.V_{>0}| - |\mathcal{G}_{\theta}.V_{>0}| \tag{3.3}$$

where $\mathcal{G}_{\theta}.V_{>0}$ denote the nodes with positive degrees.

Example 3. *Consider the same graph as Example 2 and a candidate set $\Theta = [1, 2, 3]$ to compute the $\mathcal{I}_{\mathsf{MI}}$ measure (number of edges) with $\varepsilon_2 = 1$. For the first candidate $\theta = 1$, as node 4 has degree 3, the edge addition algorithm would truncate 2 edges, for $\theta = 2$, 1 edge would be truncated and for $\theta = 3$, no edges would be truncated. We can, therefore, compute each term of the quality function for each θ given in Table 3.3. For this example, we see that $\theta = 1$ has the best quality even if it truncates the most number of edges as the error from Laplace noise overwhelms the bias error.*

We summarize the basic EM for the selection of the bounded degree in Algorithm 3. This algorithm has a complexity of $O(|\Theta|m)$, where m is the edge size of the graph, as computing the quality function for each θ candidate requires running the edge addition algorithm once. The overall Algorithm 2 has a complexity of $O(|\Sigma|n^2 + |\Theta|m)$, where the first term is due to the construction of the graph.

Privacy analysis. The privacy guarantee of Algorithm 2 depends on the budget spent for the exponential mechanism and the Laplace mechanism, as summarized below.

Theorem 2. *Algorithm 2 satisfies $(\varepsilon_1 + \varepsilon_2)$-node DP for $\mathcal{G}_{\Sigma}^{D}$ and $(\varepsilon_1 + \varepsilon_2)$-DP for the input database D.*

Proof. The proof is straightforward using the composition property of DP Proposition 1. □

The analysis is based on the sequential composition of two DP mechanisms. As stated below, we just need to analyze the sensitivity of the quality function in the exponential mechanism and the sensitivity of the measure over the projected graph.

Lemma 1. *The sensitivity of the quality function* $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ *in Algorithm 3 defined in Equation (3.1) is* $\theta_{\max} = \max(\Theta)$.

Proof. We prove the lemma for the $\mathcal{I}_{\mathsf{MI}}$ measure and show that it is similar for $\mathcal{I}_{\mathsf{P}}$. Let us assume that $\mathcal{G}$ and $\mathcal{G}'$ are two neighbouring graphs and $\mathcal{G}'$ has one extra node v^*.

$$
\begin{aligned}
&\|q_{\varepsilon_2}(\mathcal{G}, \theta) - q_{\varepsilon_2}(\mathcal{G}', \theta)\| \leq -|\mathcal{G}_{\theta_{\max}}.E| + |\mathcal{G}_{\theta}.E| - \sqrt{2}\frac{\theta}{\varepsilon_1} \\
&+ |\mathcal{G}'_{\theta_{\max}}.E| - |\mathcal{G}'_{\theta}.E| + \sqrt{2}\frac{\theta}{\varepsilon_1} \\
&\leq \left(|\mathcal{G}'_{\theta_{\max}}.E| - |\mathcal{G}_{\theta_{\max}}.E|\right) - \left(|\mathcal{G}'_{\theta}.E| - |\mathcal{G}_{\theta}.E|\right) \\
&\leq \theta_{\max} - \left(|\mathcal{G}'_{\theta}.E| - |\mathcal{G}_{\theta}.E|\right) \leq \theta_{\max}
\end{aligned}
$$

The second last inequality is due to Lemma 2 that states that $|\mathcal{G}'_{\theta_{max}}.E| - |\mathcal{G}_{\theta_{max}}.E| \leq \theta_{max}$. The last inequality is because $|\mathcal{G}'_{\theta}.E| \geq |\mathcal{G}_{\theta}.E|$. Note that the neighboring graph $\mathcal{G}'$ contains all edges of $\mathcal{G}$ plus extra edges of the added node v^*. Due to the stable ordering of edges in the edge addition algorithm, each extra edge of v^* either substitutes an existing edge or is added as an extra edge in $\mathcal{G}_{\theta}$. Therefore, the total edges $|\mathcal{G}'_{\theta}.E|$ is equal or larger than $|\mathcal{G}_{\theta}.E|$. We elaborate this detail further in the proof for Lemma 2. For the $\mathcal{I}_{\mathsf{P}}$ measure, the term in the last inequality changes to $|\mathcal{G}'_{\theta}.V_{>0}| - |\mathcal{G}_{\theta}.V_{>0}|$ and is also non-negative because $\mathcal{G}'$ contains an extra node that can only add and not subtract from the total number of nodes with positive degree. □

Lemma 2. *The sensitivity of* $f \circ \pi_{\theta}(\cdot)$ *in Algorithm 2 is* θ*, where* π_{θ} *is the edge addition algorithm with the user input* θ*, and* $f(\cdot)$ *return returns edge count for* $\mathcal{I}_{\mathsf{MI}}$ *and the number of nodes with positive degrees for* $\mathcal{I}_{\mathsf{P}}$.

Proof. Let's assume without loss of generality that $\mathcal{G}' = (V', E')$ has an additional node v^+compared to $\mathcal{G} = (V, E)$, i.e., $V' = V \cup \{v^+\}$, $E' = E \cup E^+$, and $E^+$is the set of all edges incident to v^+in $\mathcal{G}'$. We prove this lemma separately for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$.

For $\mathcal{I}_{\mathsf{MI}}$ Let Λ' be the stable orderings for constructing $\pi_\theta(\mathcal{G}')$, and t be the number of edges added to $\pi_\theta(\mathcal{G}')$ that are incident to v^+. Clearly, $t \leq \theta$ because of the θ-bounded algorithm. Let $e'_{\ell_1}, \ldots, e'_{\ell_t}$ denote these t edges in their order in Λ'. Let Λ_0 be the sequence obtained by removing from Λ' all edges incident to v^+, and Λ_k, for $1 \leq k \leq t$, be the sequence obtained by removing from Λ' all edges that both are incident to $v^+$and come after e'_{ℓ_k} in Λ'. Let $\pi_\theta^{\Lambda_k}(\mathcal{G}')$, for $0 \leq k \leq t$, be the graph reconstructed by trying to add edges in Λ_k one by one on nodes in $\mathcal{G}'$, and λ_k be the sequence of edges from Λ_k that are actually added in the process. Thus λ_k uniquely determines $\pi_\theta^{\Lambda_k}(\mathcal{G}')$; we abuse the notation and use λ_k to also denote $\pi_\theta^{\Lambda_k}(\mathcal{G}')$. We have $\lambda_0 = \pi_\theta(\mathcal{G})$, and $\lambda_t = \pi_\theta(\mathcal{G}')$.

In the rest of the proof, we show that $\forall k$ such that $1 \leq k \leq t$, at most 1 edge will differ between λ_k and λ_{k-1}. This will prove the lemma because there are at most t (upper bounded by θ) edges that are different between λ_t and λ_0.

To prove that any two consecutive sequences differ by at most 1 edge, let's first consider how the sequence λ_k differs from λ_{k-1}. Recall that by construction, Λ_k contains one extra edge in addition to Λ_{k-1} and that this edge is also incident to v^*. Let that additional differing edge be $e'_{\ell_k} = (u_j, v^+)$. In the process of creating the graph $\pi_\theta^{\Lambda_k}(\mathcal{G}')$, each edge will need a decision of either getting added or not. The decisions for all edges coming before e'_{ℓ_k} in Λ' must be the same in both λ_k and λ_{k-1}. Similarly, after e'_{ℓ_k}, the edges in Λ_k and Λ_{k-1} are exactly the same. However, the decisions for including the edges after e'_{ℓ_k} may or may not be the same. Assuming that there are a total of $s \geq 1$ different decisions, we will observe how the additional edge e'_{ℓ_k} makes a difference in decisions.

When $s = 1$, the only different decision must be regarding differing edge $e'_{\ell_k} = (u_j, v^+)$ and that must be including that edge in the total number of edges for λ_k. Also note that due to this addition, the degree of u_j gets added by 1 which did not happen for λ_{k-1}. When $s > 1$, the second different decision must be regarding an edge incident to u_j and that is because degree of u_j has reached θ, and the last one of these, denoted by $(u_j, u_{i\theta})$ which was added in λ_{k-1}, cannot be added in λ_k. In this scenario, u_j has the same degree (i.e., θ) in both λ_k and λ_{k-1}. Now if s is exactly equal to 2, then the second different decision must be not adding the edge $(u_j, u_{i\theta})$ to λ_k. Again, note here that as $(u_j, u_{i\theta})$ was not added in λ_k but was added in λ_{k-1}, there is still space for one another edge of $u_{i\theta}$. If $s > 2$, then the third difference must be the addition of an edge incident to $u_{i\theta}$ in λ_k. This process goes on for each different decision in λ_k and λ_{k-1}. Since the total number of different decisions s is finite, this sequence of reasoning will stop with a difference of at most 1 in the total number of the edges between λ_{k-1} and λ_k.

For $\mathcal{I}_\mathsf{P}$ Assume, in the worst case, the graph $\mathcal{G}$ is a star graph with n nodes such that there exists an internal node that is connected to all other $n-1$ nodes. In this scenario, there are no nodes that have 0 degrees, and the $\mathcal{I}_\mathsf{P}$ measure $= n-0 = n$. If the neighboring graph $\mathcal{G}'$ differs on the internal node, all edges of the graph are removed, and the $\mathcal{I}_\mathsf{P} = 0$. The edge addition algorithm π_θ would play a minimal role here as θ could be equal to n. □

Utility analysis. The utility of Algorithm 2 is directly encoded by the quality function of the exponential mechanism in Algorithm 3. We first define the best possible quality function value for a given database and its respective graph as

$$q_{\text{opt}}(D, \varepsilon_2) = \max_{\theta \in \Theta} q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta) \tag{3.4}$$

and the set of degree values that obtain the optimal quality value as

$$\Theta_{\text{opt}} = \{\theta \in \Theta : q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta) = q_{\text{opt}}(D, \varepsilon_2)\}. \tag{3.5}$$

However, we define e_{bias} as the difference in the number of edges or nodes in the projected graph $\mathcal{G}_\theta$ compared to that of $\mathcal{G}_{\theta_{\max}}$, instead of $\mathcal{G}$. This is to limit the sensitivity of the quality function. To compute the utility, we slightly modify the quality function without affecting the output of the exponential mechanism.

$$\tilde{q}_{\varepsilon_2}(\mathcal{G}, \theta) = q_{\varepsilon_2}(\mathcal{G}, \theta) + f(\mathcal{G}_{\theta_{\max}}) - f(\mathcal{G}_\Sigma^D), \tag{3.6}$$

where $f(\cdot)$ returns edge count for $\mathcal{I}_{\mathsf{MI}}$ and the number of nodes with positive degrees for $\mathcal{I}_\mathsf{P}$. This modified quality function should give the same set of degrees Θ_{opt} with optimal values equal to

$$\tilde{q}_{\text{opt}}(D, \varepsilon_2) = \max_{\theta \in \Theta} q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta) + f(\mathcal{G}_{\Sigma\,\theta_{\max}}^D) - f(\mathcal{G}_\Sigma^D). \tag{3.7}$$

Then, we derive the utility bound for Algorithm 2 based on the property of the exponential mechanism as follows.

Theorem 3. *On any database instance D and its respective conflict graph $\mathcal{G}_\Sigma^D$, let o be the output of Algorithm 2 with Algorithm 3 over D. Then, with a probability of at least $1-\beta$, we have*

$$|o - a| \leq -\tilde{q}_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}| \cdot \beta}) \tag{3.8}$$

where a is the true inconsistency measure over D and $\beta \leq \frac{1}{e^{\sqrt{2}}}$.

Proof. By the utility property of the exponential mechanism [121], with at most probability $\beta/2$, Algorithm 3 will sample a bad θ^* with a quality value as below

$$q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta^*) \leq q_{\text{opt}}(D, \varepsilon_2) - \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta}) \tag{3.9}$$

which is equivalent to

$$e_{\text{bias}}(\mathcal{G}, \theta^*) \geq -q_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta}) - \frac{\sqrt{2}\theta^*}{\varepsilon_2}. \tag{3.10}$$

With probability $\beta/2$, where $\beta \leq \frac{1}{e^{\sqrt{2}}}$, we have

$$\text{Lap}(\frac{\theta^*}{\varepsilon_2}) \geq \frac{\ln(1/\beta)\theta^*}{\varepsilon_2} \geq \frac{\sqrt{2}\theta^*}{\varepsilon_2} \tag{3.11}$$

Then, by union bound, with at most probability β, we have

$$\begin{aligned}
& |o - a| \\
= \; & |f(\mathcal{G}_{\theta^*}) + \text{Lap}(\frac{\theta^*}{\varepsilon_2}) - a| \\
\geq \; & a - f(\mathcal{G}_{\theta^*}) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
= \; & f(\mathcal{G}) - f(\mathcal{G}_{\theta^*}) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
= \; & f(\mathcal{G}) - f(\mathcal{G}_{\theta_{\max}}) + f(\mathcal{G}_{\theta_{\max}}) - f(\mathcal{G}_{\theta^*}) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
= \; & f(\mathcal{G}) - f(\mathcal{G}_{\theta_{\max}}) + e_{\text{bias}}(\mathcal{G}, \theta^*) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
\geq \; & -q_{\text{opt}}(D, \varepsilon_2) + f(\mathcal{G}) - f(\mathcal{G}_{\theta_{\max}}) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta}) \\
= \; & -\tilde{q}_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta})
\end{aligned} \tag{3.12}$$

□

This theorem indicates that the error incurred by Algorithm 2 with Algorithm 3 is directly proportional to the log of the candidate size $|\Theta|$ and the sensitivity of the quality

function. Without prior knowledge about the graph, $\theta_{\max}$ is usually set as the number of nodes n, and Θ includes all possible degree values up to n, resulting in poor utility. Fortunately, for our use case, the edges in the graph arise from the DCs that are available to us. In the next section, we show how we can leverage these constraints to improve the utility of our algorithm by truncating candidates in the set Θ.

3.4.2 Optimized Parameter Selection

Our developed strategy to improve the parameter selection includes two optimization techniques. The overarching idea behind these optimizations is to gradually truncate large candidates from the candidate set Θ based on the density of the graph. For example, we observe that the Stock dataset [137] has a sparse conflict graph, and its optimum degree for graph projection is in the range of $10^0 - 10^1$. In contrast, the graph for the Adult dataset sample [21] is extraordinarily dense and has an optimum degree θ greater than 10^3, close to the sampled data size. Removing unneeded large candidates, especially those greater than the true maximum degree of the graph, can help the high sensitivity issue of the quality function and improve our chances of choosing a better bound.

Our first optimization estimates an upper bound for the true maximum degree of the conflict graph and removes candidates larger than this upper bound from the initial candidate set. The second optimization is a hierarchical exponential mechanism that utilizes two steps of exponential mechanisms. The first output, θ^1, is used to truncate Θ further by removing candidates larger than θ^1 from the set, and the second output is chosen as the final candidate θ^*. In the rest of this section, we dive deeper into the details of these optimizations and discuss their privacy analysis.

Estimating the degree upper bound using FDs. Given a conflict graph $\mathcal{G}(V, E)$, we use $d(\mathcal{G}, v)$ to denote the degree of the node $v \in V$ in $\mathcal{G}$ and $d_{\max}(\mathcal{G}) = \max_{v \in V} d(\mathcal{G}, v)$ to denote the maximum degree in $\mathcal{G}$. We estimate $d_{\max}$ by leveraging how conflicts were formed for its corresponding dataset D under Σ.

The degree for each vertex in $\mathcal{G}$ can be found by going through each tuple t in the underlying dataset D and checking for how many tuples jointly with t violate the constraint set Σ. However, computing this value for each tuple is computationally expensive and highly sensitive, making it impossible to learn directly with differential privacy. We observe that the conflicts that arise due to functionality dependencies (FDs) depend on the values of the left attributes in the FD.

Example 4. *Consider the same setup as Example 2 and an FD σ : Capital $\rightarrow$ Country. We can see that the number of violations added due to the erroneous grey row is 3. This*

number is also one smaller than the maximum frequency of values occurring in the Capital attribute, and the most frequent value is "Ottawa".

Based on this observation, we can derive an upper bound for the maximum degree of a conflict graph if it involves only FDs, and this upper bound has a lower sensitivity. We show the upper bound in Lemma 3 for one FD first and later extend for multiple FDs.

Lemma 3. *Given a database D and a FD $\sigma : X \to Y$ as the single constraint, where $X = \{A_1, \ldots, A_k\}$ and Y is a single attribute. For its respective conflict graph $\mathcal{G}^D_{\Sigma=\{\sigma\}}$, simplified as $\mathcal{G}^D_\sigma$, we have the maximum degree of the graph $d_{\max}(\mathcal{G}^D_\sigma)$ upper bounded by*

$$d_{\text{bound}}(D, X) = \max_{\vec{a_X} \in \texttt{dom}(A_1) \times \ldots \times \texttt{dom}(A_k)} \text{freq}(D, \vec{a_X}) - 1, \tag{3.13}$$

where $\text{freq}(D, \vec{a_X})$ is the frequency of values $\vec{a_X}$ occurring for the attributes X in the database D. The sensitivity for $d_{\text{bound}}(D, X)$ is 1.

Proof. An FD violation can only happen to a tuple t with other tuples t' that share the same values for the attributes X. Let $\vec{a_X}^*$ be the most frequent value for X in D, i.e.,

$$\vec{a_X}^* = \text{argmax}_{\vec{a_X} \in \texttt{dom}(A_1) \times \ldots \times \texttt{dom}(A_k)} \text{freq}(D, \vec{a_X}).$$

In the worst case, a tuple t has the most frequent value $\vec{a_X}^*$ for X but has a different value in Y with all the other tuples with $X = \vec{a_X}^*$. Then the number of violations involved by t is $\text{freq}(D, \vec{a_X}^*) - 1$.

Adding a tuple or removing a tuple to a database will change, at most, one of the frequency values by 1. Hence, the sensitivity of the maximum frequency values is 1. □

Now, we will extend the analysis to multiple FDs.

Theorem 4. *Given a database D and a set of FDs $\Sigma = \{\sigma_1, \ldots, \sigma_l\}$, for its respective conflict graph $\mathcal{G}^D_\Sigma$, we have the maximum degree of the graph $d_{\max}(\mathcal{G}^D_\Sigma)$ upper bounded by*

$$d_{\text{bound}}(D, \Sigma) = \sum_{(\sigma: X \to Y) \in \Sigma} d_{\text{bound}}(D, X) \tag{3.14}$$

Proof. By Lemma 3, for each FD $\sigma : X \to Y$, a tuple may violate at most $d_{\text{bound}}(D, X)$ number of tuples. In the worst case, the same tuple may violate all FDs. □

We will spend some privacy budget ε_0 to perturb the upper bound $d_{\text{bound}}(D, X)$ for all FDs with LM and add them together. Each FD is assigned with $\varepsilon_0 / |\Sigma_{\text{FD}}|$, where Σ_{FD} is

the set of FDs in Σ. We denote this perturbed upper bound as $\tilde{d}_{\text{bound}}$ and add it to the candidate set Θ if absent.

Extension to general DCs. The upper bound derived in Theorem 4 only works for FDs but fails for general DCs. General DCs have more complex operators, such as "greater/smaller than," in their formulas. Such inequalities require the computation of tuple-specific information, which is hard with DP. For example, consider the DC $\sigma : \neg(t_i[gain] > t_j[gain] \wedge t_i[loss] < t_j[loss])$ saying that if the gain for tuple t_i is greater than the gain for tuple t_j, then the loss for t_i should also be greater than t_j. We can observe that similar analyses for FDs do not work here as the frequency of a particular domain value in D does not bound the number of conflicts related to a tuple. Instead, we have to iterate each tuple t's gain value and find how many other tuples t's violate this gain value. In the worst case, such a computation may have a sensitivity equal to the data size. Therefore, estimation using DCs may result in much noise, especially when the dataset has fewer conflicts, and the noise is added to correspond to the large sensitivity.

Our experimental study (Section 3.6) shows that datasets with general DCs have dense conflict graphs, which favors larger θs for graph projection. Hence, if we learn a small noisy upper bound $\tilde{d}_{\text{bound}}$ based on the FDs with LM, we will first prune all degree candidates smaller than $\tilde{d}_{\text{bound}}$, but then include $|V|$, which corresponds to the case when no edges are truncated, and Laplace mechanism is applied with the largest possible sensitivity $|V|$, i.e.,

$$\Theta' = \{\theta \in \Theta \mid \theta \leq \tilde{d}_{\text{bound}}\} \cup \{|V|\}. \tag{3.15}$$

Though the maximum value in Θ' is $|V|$, the sensitivity of the quality function over the candidate set Θ' remains $\tilde{d}_{\text{bound}}$. For the $|V|$ candidate, the quality function only depends on the Laplace error $\frac{\sqrt{2}|V|}{\varepsilon_2}$ and has no error from e_{bias} as no edges will be truncated.

In practice, one may skip this upper bound calculation process and skip directly to the two-step exponential mechanism if it is known that the graph is too dense or contains few FDs and more general DCs. We discuss this in detail in the experiments section.

Hierarchical EM. The upper bound d_{bound} may not be tight as it estimates the maximum degree in the worst case. The graph would be sparse with low degree values, and there is still room for pruning. To further prune candidate values in the set Θ, we use a hierarchical EM that first samples a degree value θ^* to prune values in Θ and then sample again another value θ^* from the remaining candidates as the final degree the graph projection. Our work uses a two-step hierarchical EM by splitting the privacy budget equally into halves. One may extend this EM to more steps at the cost of breaking their privacy budget more times, but in practice, we notice that a two-step is enough for a reasonable estimate.

θ	q	EM	2-EM ($\theta_1^* = \theta_3$)	2-EM ($\theta_1^* = \theta_2$)
1	-3.41	0.35	0.51	1
2	-3.82	0.33	0.49	-
3	-4.24	0.31	-	-

Table 3.4: Probabilities of choosing candidates with exponential mechanism (EM) vs two-step hierarchical exponential mechanism (2-EM). θ_1^* refers to the output of the first step of 2-EM.

Algorithm 4: Optimized EM for parameter selection

Data: Graph $\mathcal{G}(V, E)$, candidate set $\Theta = \{1, \ldots, |V|\}$, quality function q, privacy budget $\varepsilon_1, \varepsilon_2$

Result: Candidate θ^*

if Σ *mainly consists of FDs* **then**
  $\varepsilon_0 \leftarrow \varepsilon_1/4$, $\varepsilon_1 \leftarrow \varepsilon_1 - \varepsilon_0$
  Compute noisy upper bound $\tilde{d}_{\text{bound}} \leftarrow \sum_{\sigma: X \to Y} (d_{\text{bound}}(D, X) + \text{Lap}(|\Sigma_{\text{FD}}|/\varepsilon_0))$
Prune candidates $\Theta \leftarrow \{\theta \in \Theta \mid \theta \leq \tilde{d}_{\text{bound}}\} \cup \{\tilde{d}_{\text{bound}}, |V|\}$
Set $\theta_{\max} \leftarrow \min(\tilde{d}_{\text{bound}}, |V|)$
for $s \in \{1, 2\}$ **do**
  For each $\theta_i \in \Theta$, compute $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ // See Equation (3.1)
  Sample θ^* with prob $\propto \exp(\frac{\frac{\varepsilon_1}{2} q_{\varepsilon_2}(\mathcal{G}, \theta_i)}{2\theta_{\max}})$
  Prune candidates $\Theta \leftarrow \{\theta \in \Theta \mid \theta \leq \theta^*\}$
  Set $\theta_{\max} \leftarrow \theta^*$
Return θ^*

Example 5. *Consider the same setup as Example 2. For this dataset, we start with* $\Theta = [1, 2, 3]$ *and the* $\theta_{\max}$ *for this setup is 3. Assume no values are pruned in the first optimization phase. We compare a single versus a two-step hierarchical EM for the second optimization step. From Table 3.4 in Example 3, we know that the* θ_1 *has the best quality. However, as the quality values are close, the probability of choosing the best candidate is similar, as shown in Table 3.4 with* $\varepsilon = 1$*. The exponential mechanism will likely choose a suboptimal candidate in such a scenario as the probabilities are close. But if a two-step exponential mechanism is used even with half budget* $\varepsilon = 0.5$*, the likelihood of choosing the best candidate* θ_1 *goes up to* 0.51 *if the first step chose* θ_3 *or* 1 *if the first step chosen* θ_2*.*

Incorporating the optimizations into the algorithm. Algorithm 4 outlines the two optimization techniques. First, we decide when to use the estimated upper bound for the

maximum degrees, for example, when the constraint set Σ mainly consists of FDs. We will spend part of the budget ε_0 from ε_1 to perturb the upper bounds $d_{\text{bound}}(D, X)$ for all FDs with Laplace mechanism and add them together (lines 1-3). The noisy upper bound $\tilde{d}_{\text{bound}}$ prunes the candidate set (line 4). We also add $|V|$ to the candidate set if there are general DCs in Σ, and then set the sensitivity of the quality function $\theta_{\max}$ to be the minimum of the noisy upper bound or $|V|$ (line 5). Then, we conduct the two-step hierarchical exponential mechanism for parameter selection (lines 6-10). Lines 7-8 work similarly to the previous exponential mechanism algorithm with half of the remaining ε_1, where we choose a θ^* based on the quality function. However, instead of using it as the final candidate, we use it to prune values in Θ and improve the sensitivity $\theta_{\max}$ for the second exponential mechanism (lines 9-10). Then, we repeat the exponential mechanism and output the sampled θ^* (line 11). Algorithm 4 has a similar complexity of $O(|\Theta|m)$ as Algorithm 3, where $|E|$ is the edge size of the graph. The overall Algorithm 2 has a complexity of $O(|\Sigma|n^2 + |\Theta|m)$.

Privacy and utility analysis. The privacy analysis of the optimizations with our algorithm depends on the analysis of three major steps: d_{bound} computation with the Laplace mechanism, the two-step exponential mechanism, and the final measure calculation with the Laplace mechanism. By sequential composition, we have Theorem 5.

Theorem 5. *Algorithm 2 with the optimized EM in Algorithm 4 satisfies* $(\varepsilon_1 + \varepsilon_2)$*-DP.*

Proof. The total privacy budget is split in two ways for optimization and measure computation. These budgets can be composed using Proposition 1. □

Lemma 4. *The sensitivity of* $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ *in the 2-step EM (Algorithm 4) defined in Equation (3.1) is* $\theta_{\max} = \min(\tilde{d}_{\text{bound}}, |V|)$ *for 1st EM step and* $\theta_{\max} = \theta^*$ *for the 2nd EM step.*

Proof. Using Lemma 1, we know that the sensitivity of quality function $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ is given by the max over all candidates in $\max(\Theta)$. For the first step of the 2-step EM, the maximum candidate apart from $|V|$ is given by $\min(\tilde{d}_{\text{bound}}, |V|)$. For the candidate $|V|$, the quality function q differs. It only depends on the Laplace error $\frac{\sqrt{2}|V|}{\varepsilon_2}$ and has no error from e_{bias} as no edges are truncated due to $|V|$. For the 2nd step of EM, we truncate all values in the set greater than the output of the first step, i.e., θ^*. Therefore, the sensitivity becomes θ^*. □

The utility analysis in Theorem 3 for Algorithm 2 with the basic EM (Algorithm 3) still applies to the optimized EM (Algorithm 4). The basic EM usually has $\theta_{\max} = |D| = |V|$ and the full budget ε_1, while the optimized EM has a much smaller $\theta_{\max}$ and slightly lower privacy budget when the graph is sparse. In practice (Section 3.6), we see significant utility

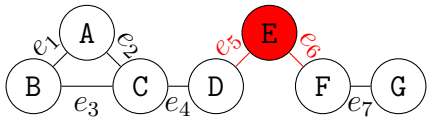


Figure 3.2: Toy graph example $\mathcal{G}$ with seven nodes (A to G) and seven edges. Consider a neighboring graph $\mathcal{G}'$ with the differing node E (red) and its two edges.

improvements by the optimized EM for sparse graphs. When the graph is dense, we see the utility degrade slightly due to a smaller budget for each EM. However, the degradation is negligible with respect to the true inconsistency measure.

3.5 DP Minimum Vertex Cover for $\mathcal{I}_{\mathsf{R}}$

This section details our approach for computing the optimal repair measure, $\mathcal{I}_{\mathsf{R}}$, using the conflict graph. $\mathcal{I}_{\mathsf{R}}$ is defined as the minimum number of vertices that must be removed to eliminate all conflicts within the dataset. For the conflict graph $\mathcal{G}_{\Sigma}^{D}$, this corresponds to finding the minimum vertex cover – an NP-hard problem. To address this, we apply a well-known polynomial-time algorithm that provides a 2-approximation for vertex cover [179]. This randomized algorithm iterates through a random ordering of edges, adding both nodes of each edge to the vertex cover if they haven't been encountered, then removes all incident edges. The process repeats until the edge list is exhausted. In our setting, we aim to compute the minimum vertex cover size while satisfying DP. A straightforward approach would be to analyze the sensitivity of the 2-approximation algorithm and add the appropriate DP noise. However, determining the sensitivity of this naive approximation is challenging, as the algorithm's output can fluctuate significantly depending on the order of selected edges. This variability is illustrated in Example 6.

Example 6. *Let us consider a graph $\mathcal{G}$ with 7 vertices* A *to* G *and 7 edges e_1 to e_7 as shown in Figure 3.2. We can have a neighboring graph $\mathcal{G}'$ by considering the vertex* E *as the differing vertex and two of its edges e_5 and e_6 as the differing edges. This example shows that according to the vanilla 2-approximate algorithm, the output for the graphs $\mathcal{G}$ and $\mathcal{G}'$ may vary drastically. For $\mathcal{G}$, if e_2 is selected followed by e_7, then the vertex cover size is 4. However, for graph $\mathcal{G}'$, if e_1 or e_4 is selected first and subsequently after the other one e_6 is selected, then the output is 6. Moreover, this difference may get significantly large if the above graph is stacked multiple times and the corresponding vertex that creates this difference is chosen every time.*

To solve the sensitivity issue, we make a minor change in the algorithm by traversing

Algorithm 5: DP approximation of minimum vertex cover size for $\mathcal{I}_{\mathsf{R}}$

Data: Graph $\mathcal{G}(V, E)$, stable global ordering Λ, privacy parameter ε
Result: DP minimum vertex cover size for $\mathcal{I}_{\mathsf{R}}$

1 Initialize vertex cover set $C = \emptyset$ and size $c = 0$
2 Initialize edge list $E_0 = E$
3 **for** $i \in \{1 \dots |\Lambda|\}$ **do**
4 pop edge $e_i = \{u, v\}$ in order from Λ
5 add u and v to C and $c = c + 2$
6 E_{i+1} = remove all edges incident to u or v from E_i
7 **Return** c + Lap$(\varepsilon/2)$

the edges in a particular order (drawing on [42]). We use a similar stable ordering Λ defined in Section 2.2.4. The new algorithm is shown in Algorithm 5. We initialize an empty vertex cover set C, its size c, and an edge list (lines 1–2). We then start an iteration over all edges in the same ordering as the stable ordering Λ (line 3). For each edge $e_i = \{u, v\} \in E$ that is part of the graph, we add both u and v to C and correspondingly increment the size c (lines 4–5). We remove all other edges, including e_i, connected to u or v from E and continue the iteration (line 5). Finally, we return the noisy size of the vertex cover (line 6). The sensitivity of this algorithm is given by Proposition 3.

Proposition 3. *Algorithm 5 obtains a vertex cover, and its size has a sensitivity of 2.*

Proof. Let's assume without loss of generality that $\mathcal{G}' = (V', E')$ has an additional node v^+compared to $\mathcal{G} = (V, E)$, i.e., $V' = V \cup \{v^+\}, E' = E \cup E^+$, and E^+ is the set of all edges incident to v^+ in $\mathcal{G}'$. We prove the theorem using a mathematical induction on i that iterates over all edges of the global stable ordering Λ.

Base: At step 0, the value of c and c' are both 0.

Hypothesis: As the algorithm progresses at each step i when the edge e_i is chosen, either the edges of graph $\mathcal{G}'$ which is denoted by E'_i has an extra vertex or the edge of graph $\mathcal{G}$ has an extra vertex. Thus, we can have two cases depending on some node v^* and its edges $\{v^*\}$. Note that at the beginning of the algorithm, v^* is the differing node v^+ and $\mathcal{G}'$ has the extra edges of v^*/v^+, but v^* may change as the algorithm progresses. The cases are as illustrated below:

- Case 1: E_i does not contain any edges incident to v^*, $E'_i = E_i + \{v^*\}$ and the vertex cover sizes at step i could be $c_i = c'_i$ or $c_i = c'_i + 2$.

- Case 2: E'_i does not contain any edges incident to v^*, $E_i = E'_i + \{v^*\}$ and the vertex cover sizes at step i could be $c_i = c'_i$ or $c'_i = c_i + 2$.

- Case 3: $E_i = E'_i$ and the vertex cover sizes at step i is $c_i = c'_i$. This case occurs only when the additional node v^+ has no edges.

Induction: At step $i+1$, lets assume an edge $e_{i+1} = \{u, v\}$ is chosen. Depending on the i^{th} step, we can have 2 cases as stated in the hypothesis.

- Case 1 (When E'_i has the extra edges of v^*): We can have the following subcases at step $i+1$ depending on e_{i+1}.

  a) If the edge is part of E'_i but not of E_i ($e_{i+1} \in E'_i \setminus E_i$): Then $e_{i+1} = \{u, v\}$ should not exist in E_i (according to the hypothesis at the i step) and one of u or v must be v^*. Let's assume without loss of generality that v is v^*. The algorithm will add (u, v) to C' and update $c'_{i+1} = c_i + 2$. Hence, we have either $c'_{i+1} = c_{i+1}$ or $c'_{i+1} = c + 2$.
  In addition, all edges of u and v/v^* will be removed from E'_{i+1}. Thus, we have $E_{i+1} = E'_{i+1} + \{u\}$, where $\{u\}$ represent edges of u. Now u becomes the new v^* and moves to Case 2 for the $i+1$ step.

  b) If the edge is part of both E'_i and E_i($e_{i+1} \in E'_i$ and $e_{i+1} \in E_i$): In this case (u, v) will be added to both C and C' and the vertex sizes with be updated as $c_{i+1} = c_i + 2$ and $c'_{i+1} = c' + 2$.
  Also, the edges adjacent to u and v will be removed from E_i and E'_i. We still have $E'_i = E + v^*$ (the extra edges of v^* and remain in case 1 for step i+1.

  c) If the edge is part of neither E'_i nor E_i (If $e_{i+1} \in E'_i$ and $e_{i+1} \in E_i$): the algorithm makes no change. The previous state keep constant: $E'_{i+1} = E'_i, E_{i+1} = E_i$ and $c'_{i+1} = c'_i, c_{i+1} = c_i$. The extra edges of v^* are still in E'_{i+1}.

- Case 2 (When E_i has the extra edges of v^*) : This case is symmetrical to Case 1. There will be three subcases similar to Case 1 – a) in which after the update the state of the algorithm switches to Case 1, b) in which the state remains in Case 2, and c) where no update takes place.

- Case 3 (When $E_i = E'_i$): In this case, the algorithm progresses similarly for both the graphs, and remains in case 3 with equal vertex covers, $c_{i+1} = c'_{i+1}$.

Our induction proves that our hypothesis is true and the algorithm starts with Case 1 and either remains in the same case or oscillates between Case 1 and Case 2. Hence as per our hypothesis statement, the difference between the vertex cover sizes are upper bounded by 2. □

Induction: At step $i+1$, lets assume an edge $e_{i+1} = \{u, v\}$ is chosen. Depending on the i^{th} step, we can have 2 cases as stated in the hypothesis.

- Case 1 (When E'_i has the extra edges of v^*): We can have the following subcases at step $i+1$ depending on e_{i+1}.

  a) If the edge is part of E'_i but not of E_i ($e_{i+1} \in E'_i \setminus E_i$): Then $e_{i+1} = \{u, v\}$ should not exist in E_i (according to the hypothesis at the i step) and one of u or v must be v^*. Let's assume without loss of generality that v is v^*. The algorithm will add (u, v) to C' and update $c'_{i+1} = c_i + 2$. Hence, we have either $c'_{i+1} = c_{i+1}$ or $c'_{i+1} = c + 2$.
  In addition, all edges of u and v/v^* will be removed from E'_{i+1}. Thus, we have $E_{i+1} = E'_{i+1} + \{u\}$, where $\{u\}$ represent edges of u. Now u becomes the new v^* and moves to Case 2 for the $i+1$ step.

  b) If the edge is part of both E'_i and E_i($e_{i+1} \in E'_i$ and $e_{i+1} \in E_i$): In this case (u, v) will be added to both C and C' and the vertex sizes with be updated as $c_{i+1} = c_i + 2$ and $c'_{i+1} = c' + 2$.
  Also, the edges adjacent to u and v will be removed from E_i and E'_i. We still have $E'_i = E + v^*$ (the extra edges of v^* and remain in case 1 for step i+1.

  c) If the edge is part of neither E'_i nor E_i (If $e_{i+1} \in E'_i$ and $e_{i+1} \in E_i$): the algorithm makes no change. The previous state keep constant: $E'_{i+1} = E'_i, E_{i+1} = E_i$ and $c'_{i+1} = c'_i, c_{i+1} = c_i$. The extra edges of v^* are still in E'_{i+1}.

- Case 2 (When E_i has the extra edges of v^*) : This case is symmetrical to Case 1. There will be three subcases similar to Case 1 – a) in which after the update, the state of the algorithm switches to Case 1, b) in which the state remains in Case 2, and c) where no update takes place.

- Case 3 (When $E_i = E'_i$): In this case, the algorithm progresses similarly for both the graphs, and remains in case 3 with equal vertex covers, $c_{i+1} = c'_{i+1}$.

Our induction proves that our hypothesis is true. The algorithm starts with Case 1, either in the same case or oscillates between Case 1 and Case 2. Hence, as per our

hypothesis statement, the difference between the vertex cover sizes is upper-bounded by 2. □

Example 7. *Let us consider our running example in Figure 3.2 as input to Algorithm 5 and use it to understand the proof. We have two graphs – $\mathcal{G}$ which has 6 vertices $V = [\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{F}, \mathtt{G}]$ and edges $E = [e_1, e_2, e_3, e_4, e_7]$ and $\mathcal{G}'$ has 7 vertices $V' = [\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}]$ and edges $E' = [e_1, e_2, \ldots, e_7]$. The total possible number of edges is $\binom{7}{2} = 21$, and we can have a global stable ordering of the edges Λ depending on the lexicographical ordering of the vertices as $e_1, e_2, e_3, \ldots, e_{21}$. When the algorithm starts, both vertex cover sizes are initialized to $c = 0, c' = 0$, and the algorithm's state is in Case 1 with $v^* = \mathtt{E}$. We delineate the next steps of the algorithm below:*

- *Iteration 1 (Subcase 1b) : $e_1(\mathtt{A}, \mathtt{B})$ is chosen. $\mathtt{A}$ and $\mathtt{B}$ are both in E_0 and E'_0. Hence $c = 2, c' = 2$.*
- *Iteration 2 and 3 (Subcase 1c) : $e_2(\mathtt{A}, \mathtt{C})$ and $e_3(\mathtt{B}, \mathtt{C})$ are chosen. Both are removed in iteration 1. Hence $c = 2, c' = 2$.*
- *Iteration 4 (Subcase 1b) : $e_4(\mathtt{C}, \mathtt{D})$ is chosen. $\mathtt{C}$ and $\mathtt{D}$ are both in E_3 and E'_3. Hence $c = 4, c' = 4$.*
- *Iteration 5 (Subcase 1c) : $e_5(\mathtt{D}, \mathtt{E})$ is chosen. It's removed from E'_4 in iteration 4 and was never present in E. Hence $c = 4, c' = 4$.*
- *Iteration 6 (Subcase 1a) : $e_6(\mathtt{E}, \mathtt{F})$ is chosen. It's in E'_5 but not in E_5. Hence, $c = 4, c' = 6$ and the new $v^* = \mathtt{E}$.*
- *Iteration 7 (Subcase 2a) : $e_7(\mathtt{F}, \mathtt{G})$ is chosen. It's in E_6 but removed from E_6 in Iteration 6. Hence, $c = 6, c' = 6$, and the algorithm is complete.*

Privacy and utility analysis. We now show the privacy and utility analysis of Algorithm 5 using Theorem 6 below.

Theorem 6. *Algorithm 5 satisfies ε-node DP and always outputs the size of a 2-approximate vertex cover of graph $\mathcal{G}$.*

Proof. The privacy analysis of Algorithm 5 is straightforward as we calculate the private vertex cover using the Laplace mechanism with sensitivity 2 according to Proposition 3. The utility analysis can be derived from the original 2-approximation algorithm. The stable ordering Λ in Algorithm 5 can be perceived as one particular random order of the edges and hence has the same utility as the original 2-approximation algorithm. □

Dataset	#Tuple	#Attrs	#DCs(#FDs)	Max Deg 1% RNoise
Adult [21]	32561	15	3 (2)	9635
Flight [136]	500000	20	13 (13)	1520
Hospital [149]	114919	15	7 (7)	793
Stock [137]	122498	7	1 (1)	1
Tax [38]	1000000	15	9 (7)	373

Table 3.5: Description of datasets. The max deg column shows the maximum degree of any node in a $10k$ rows subset of the conflict graph of the dataset with 1% RNoise.

3.6 Experiments

This section presents our experiment results on computing the three measures outlined in Section 3.4 and Section 3.5. The questions that we ask through our experiments are as follows:

1. How far are the private measures from the true measures?
2. How do the different strategies for the degree truncation bound compare against each other?
3. How do our methods perform with different privacy budgets?

3.6.1 Experimental Setup

All our experiments are performed on a server with Intel Xeon Platinum 8358 CPUs (2.60GHz) and 1 TB RAM. Our code is in Python 3.11 and can be found in the artifact submission. All experiments are repeated for 10 runs, and the mean error value is reported.

Datasets and violation generation.

We replicate the exact setup as Livshits et al. [110] for experimentation. We conduct experiments on five real-life datasets and their corresponding DCs as described in Table 3.5.

- Adult [21]: Annual income results from various factors.
- Flight [136]: Flight information across the US.

- Hospital [149]: Information about different hospitals across the US and their services.
- Stock [137]: Trading stock information on various dates
- Tax [38]: Personal tax infomation.

These datasets are initially consistent with the constraints. All experiments are done on a subset of $10k$ rows, and violations are added similarly using both their proposed algorithms, namely CONoise (for Constraint-Oriented Noise) and RNoise (for Random Noise).

CONoise introduces random violations of the constraints by running 200 iterations of the following procedure:

1. Randomly select a constraint σ from the constraint set Σ.
2. Randomly select two tuples t_i and t_j from the database.
3. For every predicate $\phi = (t_i[a_1] \circ t_j[a_2])$ of σ:
   - If t_i and t_j jointly satisfy ϕ, continue to the next predicate.
   - If $\circ \in \{=, \leq, \geq\}$, change either $t_i[a_1]$ or $t_j[a_2]$ or vice versa (the choice is random).
   - If $\circ \in \{<, >, \neq\}$, change either $t_i[a_1]$ or $t_2[a_2]$ (the choice is again random) to another value from the active domain of the attribute such that ϕ is satisfied, if such a value exists, or a random value in the appropriate range otherwise.

The second algorithm, RNoise, is parameterized by the parameter α that controls the noise level by modifying α of the values in the dataset. At each iteration of RNoise, we randomly select a database cell corresponding to an attribute that occurs in at least one constraint. Then, we change its value to another value from the active domain of the corresponding attribute (with probability 0.5) or a typo.

The datasets vary immensely in the density of their conflict graphs as described in the max degree column of Table 3.5. For example, the Adult $10k$ nodes subset has a maximum degree of 9635, whereas the Stock dataset has a maximum of 1 with the same amount of conflict addition.

Metrics. Following Livshits et al. [110], we randomly select a subset of 10k rows of each dataset, add violations to the subset, and compute the inconsistency measures on the dataset with violations. To measure performance, we utilize the normalized ℓ_1 distance error [52], $|\mathcal{I}(D, \Sigma) - \mathcal{M}(D, \Sigma, \varepsilon)|/\mathcal{I}(D, \Sigma)$, where $\mathcal{M}(D, \Sigma, \varepsilon)$ represents the estimated

private value of the measure and $\mathcal{I}(D, \Sigma)$ denotes the true value. For the $\mathcal{I}_{\mathsf{R}}$ measure, we use the linear approximation algorithm from Livshits et al. [110] to estimate the non-private value.

Algorithm variations. We experiment with multiple different variations of Algorithm 2 for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$. The initial candidate set for the degree bound is $\Theta = [1, 5, 10, 100, 500, 1000, 2000, 3000, \ldots$ with multiples of 1000 along with some small candidates.

- *Baseline 1*: naively sets the bound θ^* to the maximum possible degree $|V|$ in Algorithm 2 by skipping line 2 and the unused privacy budget ε_1 is used for the final noise addition step.
- *Baseline 2*: sets the bound θ^* to the actual maximum degree of the conflict graph $d_{\max}(\mathcal{G}_\Sigma^D)$. Note that this is a non-private baseline that only acts as an upper bound and is one of the best values that can be achieved without considering privacy constraints.
- *Exponential mechanism*: choose θ^* over the complete candidate set Θ using the basic EM in Algorithm 3.
- *Hierarchical exponential mechanism*: chooses θ^* using a two-step EM with an equal budget for each step in Algorithm 4, but skipping Lines 1-5 of the upper bound computation step.
- *Upper bound + hierarchical exponential mechanism (our approach)*: encompasses both the optimization strategies, including the upper bound computation and the hierarchical exponential mechanism discussed in Section 3.4.2 (the full Algorithm 4).

By default, we experiment with a total privacy budget of $\varepsilon = 1$ unless specified otherwise.

3.6.2 Results

True vs private estimation

In Figures 3.3 and 3.4, we plot the true vs. private estimates at $\varepsilon = 1$ for all datasets with RNoise ($\alpha = 0.01$) and CONoise (200 iterations) respectively. The datasets are ordered according to their densities from left to right. Each figure contains the measured value ($\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_{\mathsf{P}}$, or $\mathcal{I}_{\mathsf{R}}$) on the Y-axis and the number of iterations on the X-axis. For the CONoise, the number of iterations is set to 200 for every dataset, and for RNoise, the iterations correspond to the number of iterations required to reach 1% ($\alpha = 0.01$) number of random violations. The orange line corresponds to the true value of the measure, and the blue line corresponds to the private measure using our approach. For $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ measures, the blue line represents the upper bound + hierarchical exponential mechanism strategy

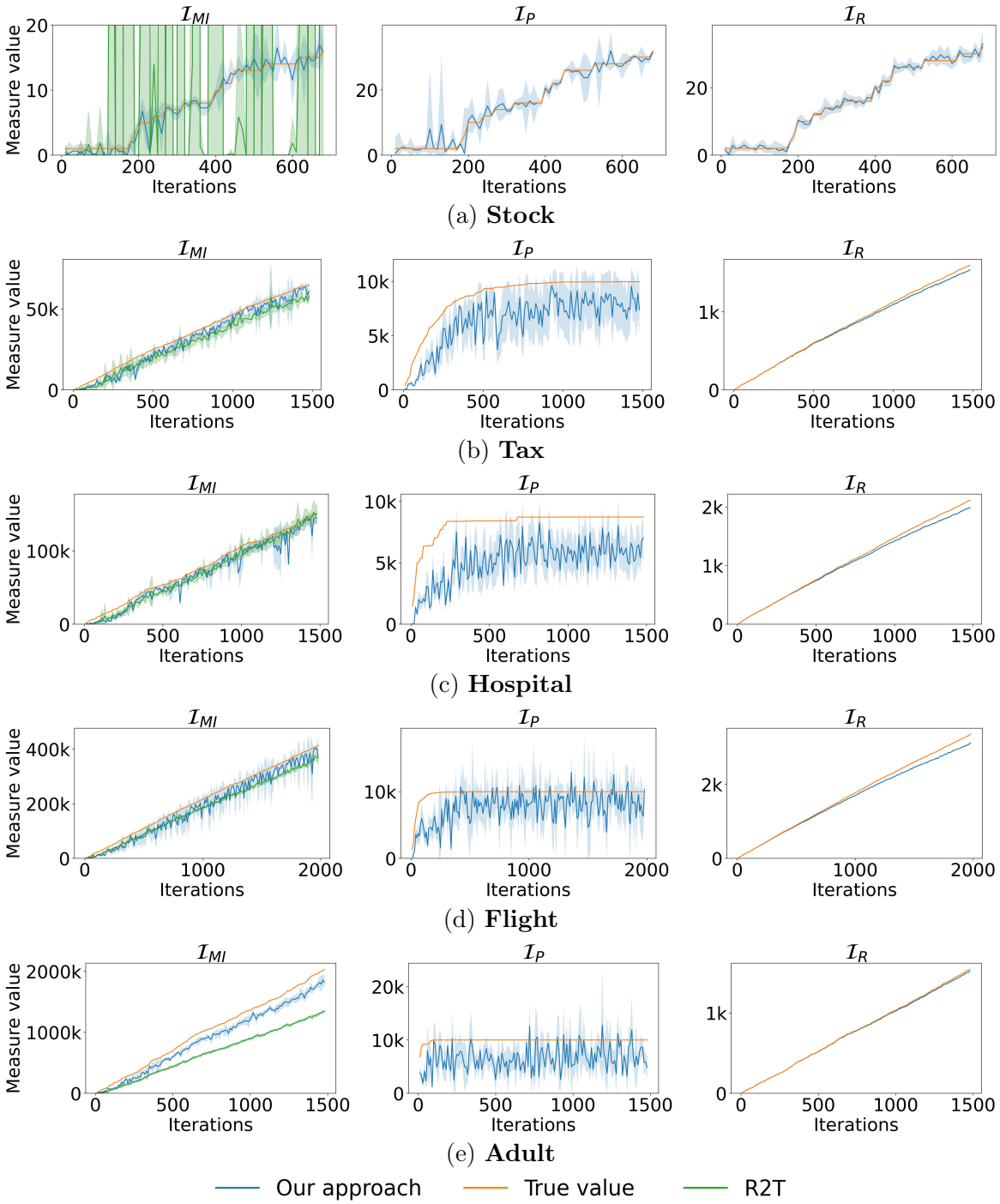


Figure 3.3: True vs Private estimates for all datasets with RNoise $\alpha = 0.01$ at $\varepsilon = 1$. Each row shows the three measures — $\mathcal{I}_{\mathsf{P}}$ (Positive degree nodes), $\mathcal{I}_{\mathsf{MI}}$ (Number of edges), and $\mathcal{I}_{\mathsf{R}}$ (Size of vertex cover) — for the corresponding dataset.

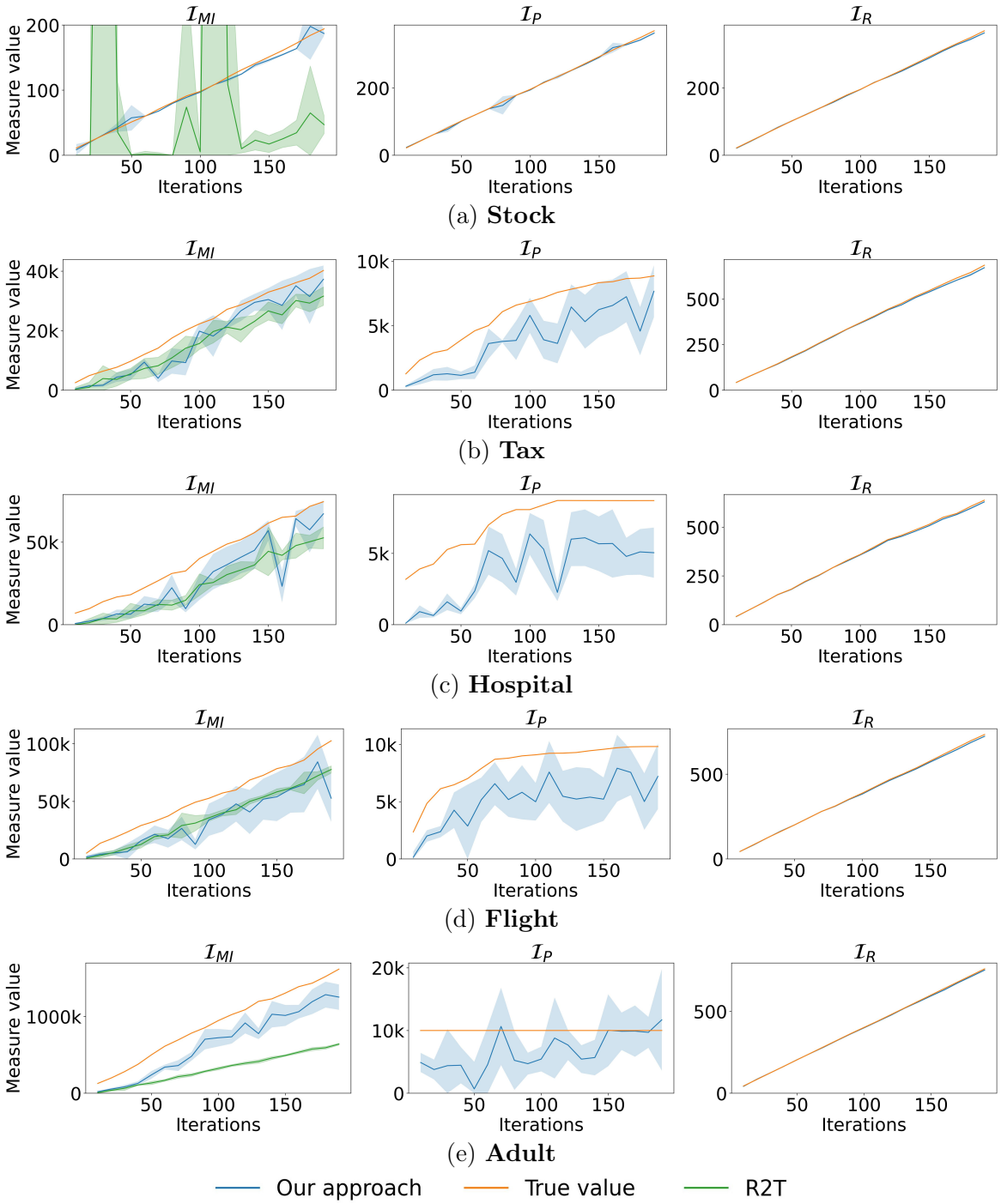
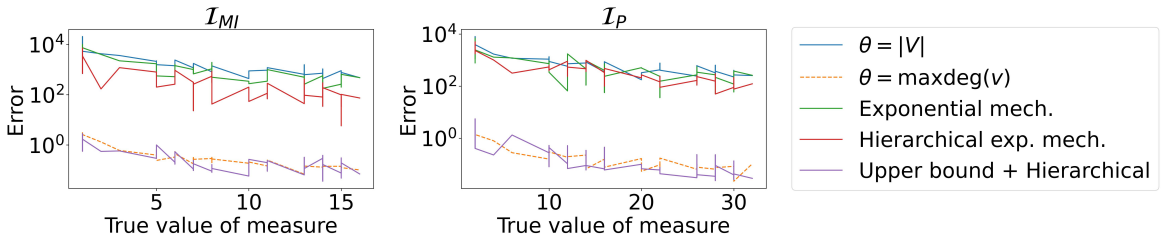
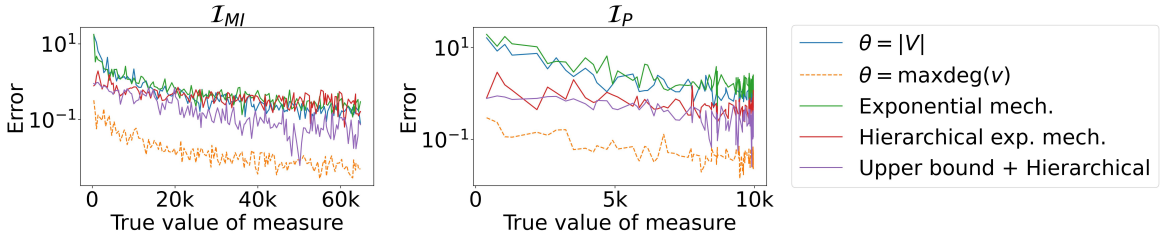


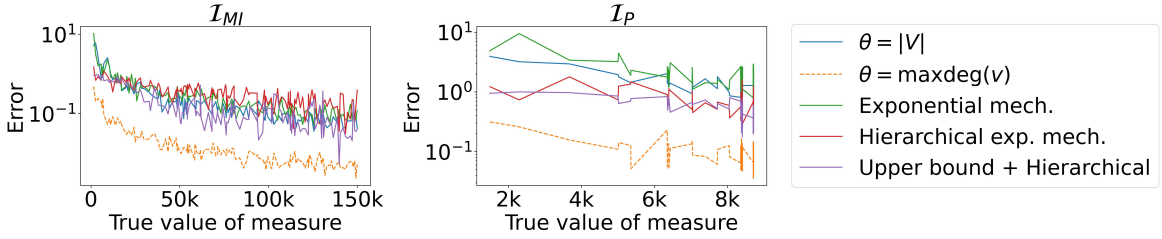
Figure 3.4: True vs Private estimates for all datasets with RNoise $\alpha = 0.01$ at $\varepsilon = 1$. Each row shows the three measures — $\mathcal{I}_{\mathsf{MI}}$ (Number of edges), $\mathcal{I}_{\mathsf{P}}$ (Positive degree nodes), and $\mathcal{I}_{\mathsf{R}}$ (Size of vertex cover) — for the corresponding dataset.



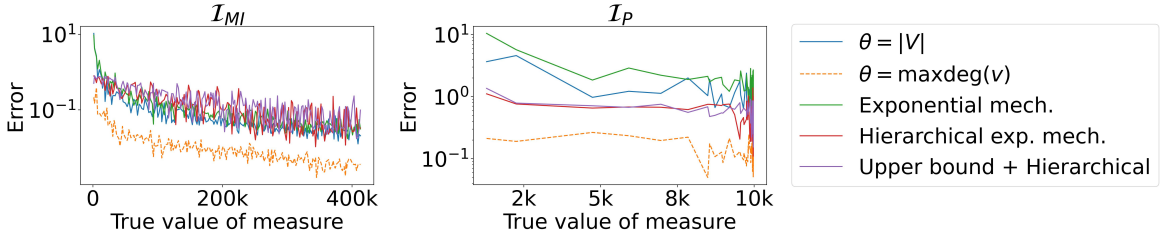
(a) **Stock**



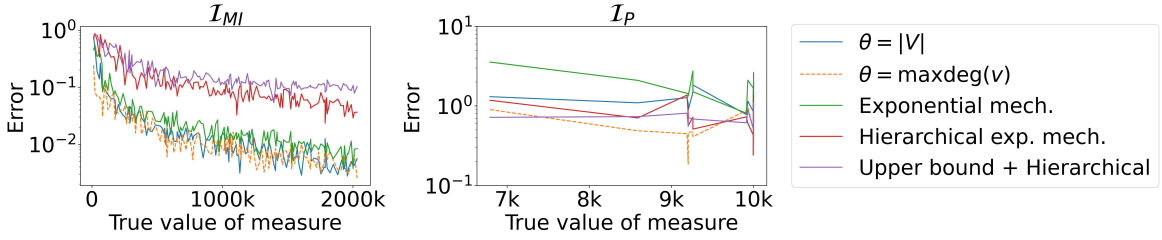
(b) **Tax**



(c) **Hospital**



(d) **Flight**



(e) **Adult**

Figure 3.5: Computing different strategies for choosing θ for all datasets with RNoise at $\alpha = 0.01$ and $\varepsilon = 1$. The datasets are arranged according to their densities from sparsest (top) to densest (bottom).

described in Section 3.4.2 along with its standard deviation in shaded blue. For the $\mathcal{I}_{\mathsf{R}}$ measure, it represents the private minimum vertex cover size algorithm. We also add a baseline approach using a state-of-the-art private SQL approach called R2T [47]. We add this baseline only for the $\mathcal{I}_{\mathsf{MI}}$ measure as $\mathcal{I}_{\mathsf{R}}$ cannot be written with SQL, and $\mathcal{I}_{\mathsf{P}}$ requires the DISTINCT/GROUP BY clause that R2T does not support. Based on the experiments, we draw three significant observations.

First, compared to the SQL baseline (R2T), our approach has a better relative error on average across all datasets. However, R2T is slightly behind for moderate to high dense datasets such as Tax (0.207 vs. 0.334), Hospital (0.209 vs. 0.386), and Flight(0.202 vs. 0.205) and Adult (0.187 vs. 0.269) but fails miserably for sparse datasets such as Stock (0.492 vs. 137.05). This is because the true value of the measure is small, and R2T adds large amounts of noise.

Second, our approach for the $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ fluctuates more and has a higher standard deviation compared to the $\mathcal{I}_{\mathsf{R}}$ measure. This is because of the privacy noise due to the relatively high sensitivity of our upper bound + hierarchical exponential mechanism approach. On the other hand, the vertex cover size approach for $\mathcal{I}_{\mathsf{R}}$ has a sensitivity equal to 2 and, therefore, does not show much fluctuation when the true measure value is large enough.

Third, we observe that our approach generally performs well across all five datasets and all inconsistency measures. The $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ measures have average errors of 0.25 and 0.46, respectively, across all datasets where Stock is the worst performing dataset for $\mathcal{I}_{\mathsf{MI}}$ and Adult is the worst performing for $\mathcal{I}_{\mathsf{P}}$. The $\mathcal{I}_{\mathsf{R}}$ performs the best with an average error of 0.08, with Stock as the worst-performing dataset. We investigate the performance of each dataset in detail in our next experiment and find out that the density of the graphs plays a significant role in the performance of our algorithms.

Comparing different strategies for choosing θ.

In Figure 3.5, we present the performance of different algorithm variations in computing the $\mathcal{I}_{\mathsf{P}}$ and $\mathcal{I}_{\mathsf{MI}}$ measures for all datasets using RNnoise at $\alpha = 0.01$ and $\varepsilon = 1$. The y-axis in each figure shows the logarithmic scaled error, while the x-axis displays the actual measure value, with different colors representing the strategies. The graphs are ordered from most sparse (Stock) to least sparse (Adult) to compare methods for choosing the θ value at $\varepsilon = 1$ ($\varepsilon_1 = 0.4$, $\varepsilon_2 = 0.6$). The methods include all variations described in the algorithm variation section. We note all the strategies are private except baseline 2 (orange dash line) that sets θ as the true maximum degree of the conflict graph.

Our experimental results, based on error trends and graph density, reveal several key observations.

First, we consistently observed that the initial error was higher at smaller iterations across all five datasets and inconsistency measures. This is because, at smaller iterations, the true value of the measures is small due to fewer violations, and the privacy noise dominates the signal of true value.

Second, for the sparsest dataset (Stock), all strategies have errors of magnitude 3-4 larger, except the non-private baseline (orange) and our approach using both upper bound and hierarchical exponential mechanism (purple). This is because the candidate set contains many large candidates, and it is crucial to prune it using the upper bound strategy to get meaningful results.

Third, for the moderately sparse graphs (Tax and Hospital), our approach consistently (purple) consistently outperformed other private methods. However, the two-step hierarchical exponential mechanism (red), which had a 3 magnitude higher error for Stock, demonstrated comparable performance within a 1-magnitude error difference for Tax and Hospital. This suggests that when the true max degree is not excessively low, estimating it without the upper bound strategy can be effective.

Finally, for the densest graphs (Flight and Adult), we observe that the optimized exponential mechanisms (red and purple) outperform the private baselines (blue and green) for the $\mathcal{I}_{\mathsf{P}}$ measure (nodes with positive degree) plots (above). However, they fail to beat even the naive baseline (blue) for the $\mathcal{I}_{\mathsf{MI}}$ (number of edges) measure (below). This is because the optimal degree bound value $\mathcal{I}_{\mathsf{MI}}$ over the dense graphs is close to the largest possible value $|V|$. For such a case, our optimized EM is not able to prune too many candidates and lower the sensitivity, and hence, it wastes some of the privacy budget in the pruning process. However, the relative errors of all the algorithms are reasonably small for dense graphs, and the noisy answers do preserve the order of the true measures (shown in previous experiments in Figures 3.3 and 3.4).

Varying privacy budget

Figure 3.6 illustrates how our proposed DP algorithms perform at varying privacy budgets with $\varepsilon \in [0.1, 0.2, 0.5, 1.0, 2.0, 3.0, 5.0]$. The rightmost figure for the repair measure $\mathcal{I}_{\mathsf{R}}$ has a log scale on the y-axis for better readability. We experiment with three datasets of different density properties (sparsest Stock and densest Flight) and show that our algorithm gracefully scales with the ε privacy budget for all three inconsistency measures. We also observe that the algorithm has a more significant error variation at smaller epsilons (< 1)

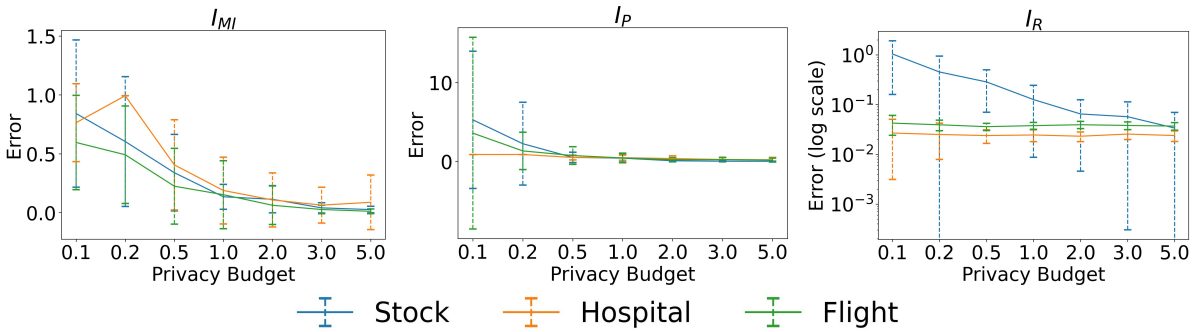


Figure 3.6: Computing inconsistency measures for different datasets with RNoise at $\alpha = 0.005$ and varying privacy budgets

except for Stock, which has a larger variation across all epsilons. This happens when the true value for this measure is small, and adding noise at a smaller budget ruins the estimate drastically. For the $\mathcal{I}_{\mathsf{R}}$ measure, the private value reaches with 0.05 error at $\varepsilon = 3$ for Stock and as early as $\varepsilon = 0.1$ for others.

Chapter 4

Data exploration and Cleaning: Generating synthetic data with missing data

4.1 Introduction

In this chapter, we show our work for working with missing data in private learning. Specifically, we look into the task of generating synthetic data from a private dataset with missing values. Generating synthetic data with differential privacy is a private learning method where a similar synthetic data is generated by quering the private dataset within the privacy budget. The synthetic dataset, once generated, can be made public, and the analyst can use it for any number of downstream tasks [35, 36, 87, 193, 19].

Despite a number of work [193, 62, 146, 87] that succeed in generating synthetic data with DP guarantees, they only look at a simple scenario where the input data has no missing values. Several prior studies [107, 66] have reported on the prevalence of missing data in various fields. For instance, a study of 9 publicly available healthcare datasets commonly used in machine learning research found that the proportion of missing values ranged from 0.2% to 78.6% [66]. The presence of missing data can be attributed to multiple reasons, such as human errors [11] and privacy regulations like GDPR [2] which allow people the "right to forget" where one may ask their data to be deleted completely [148].

In our work, we formally define the research problem of generating synthetic data for sensitive data with missing values using DP. We consider a missing mechanism that takes complete ground truth data and outputs data with missing values. Under this setup, we can offer DP to either *the incomplete data* or *the ground truth data*. For each privacy guarantee, we study how to handle missing data in the synthetic data generation process. While it is desired to have a unified comprehensive approach to solve the problem, we identify multiple challenges in achieving such a solution. First, there is no straightforward winning algorithm for DP synthetic data generation even without missing data as demonstrated by prior benchmarking work [168]. Second, the design space for a solution is huge for dealing with missing data. We explore several techniques, including the vanilla approach that uses complete rows only [131], common imputation techniques (e.g., statistical methods, machine learning methods) [107, 155, 133, 102], and other differentially private imputation methods [41, 62]. We show that all these methods have their limitations, such as discarding too many rows or incurring high privacy costs, leading to poor-quality synthetic data. Therefore in our work we present a comprehensive list of possible approaches to solve the problem including two vanilla approaches and three adaptive approaches, as a contribution to understanding the design space for this new problem.

In addition, we show the relationship between the privacy guarantee for these DP mechanisms that they offer for the incomplete data and that for the ground truth data. To do so, we model the missing mechanism as a sampling process and obtain a tighter upper bound for the privacy loss to the ground truth data via sampling amplification techniques [17, 163]. Unlike prior work for sampling amplification that considers a random subset, we make use of the randomness due to missing values to amplify the privacy for ground truth data.

4.2 Preliminaries on Missing Data

For missing data, we define a missingness indicator matrix $M = [\ldots, m_{ij}, \ldots]$ of size $n \times k$, where $i \in [1, n], j \in [1, k]$ and, shorthand m_i to point to i^{th} row of M. Each entry of M has one-to-one relation with D such that, $m_{ij} = 1$ if D_{ij} is missing and $m_{ij} = 0$ otherwise.

Missing data is classified into different types using missing mechanisms. A missing mechanism $M_\phi : \mathcal{D} \rightarrow \mathcal{D}$ takes as input the ground truth dataset $\bar{D}$ and outputs an incomplete dataset D. It is parameterized by Φ, a set of probabilities, which refers to the set of probabilities that control the unknown missing data process. Three missing types can be defined using Φ and the conditional distribution of missing indicator m_i given the dataset D_i [107, 155].

Missing completely at random (MCAR) assumes the probability of missingness is completely independent of the data. Under MCAR, any two rows of the dataset, regardless of their values, for the same attribute have the same probability of having a missing value. Hence, the parameter set Φ consists of $\{\phi_j | j \in [1, k]\}$, where ϕ_j is the probability of any row having a missing value for the jth column. For $j \in [1, k]$ and $i \in [1, n]$, $\Pr[m_{ij}|D] = \Pr[m_{ij}] = \phi_j$. Hence, the probability of a row having no missing values is $\prod_{j=1}^{k}(1 - \phi_j)$.

Missing at random (MAR) captures the scenario where the probability of missingness is independent of the missing values given the observed data. In other words, under MAR how likely a value is to be missing can be estimated based on the non-missing data. Consider examples, 1) Young people have missing IQ (because they haven't taken an IQ test yet), and MAR models the same probability of missing IQ attribute for rows of the same age, regardless of their IQ values; 2) Businessmen are less likely to share their income, and MAR models the same probability for income values for rows that have an occupation as 'Business.' MAR, therefore, is parameterized by a set of conditional probabilities Φ where each $\phi \in |\Phi|$ maps relationships between observed and missing values in the dataset. For $\phi_x \in \Phi$ that models the jth column's missingness, and for $i \in [1, n]$, $\Pr[m_{ij}|D_{(0)}] = \Pr[m_{ij}|D_{(0)}, D_{ij}] = \phi_x$, where $D_{(0)}$ refers to the observed attributes in the dataset.

Missing not at random (MNAR) captures the scenario when given all the observed information, the probability of missingness depends on any other unobserved missing values in the dataset. Consider examples, 1) Students with missing attendance also have missing scores, and the probability of the missing scores depending on missing attendance 2) People who smoke don't want to mention they smoke. Here MNAR models the probability of missing smokers based on the attribute of smoking. With MNAR missingness, Φ consists of a set of conditional probabilities that map the probability of an attribute to be missing given its own value.

Datasets often contain various missing data types, but identifying them without domain knowledge remains challenging [105, 106]. While statistical tests exist for MCAR cases (e.g., Little's MCAR test, pattern mixture models) [105, 106], identifying MAR and MNAR remains unsolved due to complex interactions between observed and unobserved variables [177]. Despite binary indicator tests based on modeling for MAR, these lack conclusiveness and are sensitive to analysis assumptions. For private datasets, these tests also need to be computed privately by using a portion of the privacy budget.

4.3 Problem Statement

Consider a private dataset behind a privacy firewall with n rows and k attributes. A trusted curator aims to generate a synthetic version of the same size with an end-to-end (ε, δ)-DP guarantee while preserving maximum utility. Real-world data collected by curators may contain missing values, a scenario overlooked in prior work. We address this by formalizing two versions of the problem based on privacy considerations. First, we offer a DP guarantee for the incomplete dataset held by the data curator.

Problem 1. *[Privacy for Incomplete Data] Consider collecting data from a ground truth data $\bar{D}$ of n rows owned by n individuals, a missing mechanism $M_\Phi : \mathcal{D} \to \mathcal{D}$ is involved that takes in $\bar{D}$ and outputs a dataset D of n rows but with missing values. A trusted data curator uses this dataset D as input and aims to generate a synthetic data D^* of n rows with a mechanism $M : \mathcal{D} \to \mathcal{D}$ such that D^* minimizes $d(f(D) - f(D^*))$, where $f : \mathcal{D} \to \mathbb{R}^l$ is a utility metric function that the user is interested in, $d(\cdot, \cdot)$ is a distance metric and M offers (ε, δ)-DP to the input data D.*

In Section 4.4, we delineate multiple options for the data curator to deal with incomplete data, put forward challenges that come with them, and discuss which option might be the best and when. The first option is a naïve adaptation of prior work for DP data generation by simply discarding the rows with missing values [155]. We refer to this approach as *complete row only*. This approach can fail in many cases, as detailed in Section 4.4.1. For instance, if all rows have some missing values, then there will be no input data for the data generation methods. A second approach is to impute the missing parts with inferred values from the observed data. We denote this approach as *imputation first approach*. However, as our data is private, the imputation process needs to be privatized as well and the additional incurred privacy cost must be accounted for in the privacy budget. In section 4.4.1, we explore the privacy costs of imputation and show how they can be expensive in practice. Rather than having separate processes for imputation and synthetic data generation, we can integrate these two processes into one. This line of thought motivates us to a new approach, which we call *adaptive recourse approach*. In Section 4.4.2, we improve upon three categories of DP generation approaches and demonstrate their effectiveness in generating synthetic data from incomplete data. These strategies use no extra privacy budget and solely improve by observing available data in the dataset.

The incomplete data D can be modeled as a sample generated from a complete ground truth dataset $\bar{D}$ via a missing mechanism M_Φ. If the privacy goal is to protect the ground truth dataset $\bar{D}$ with DP guarantee, how will the problem and the solution be different? We formalize the second problem as follows.

Problem 2. *[Privacy for Ground Truth Data] Consider the same setup as Problem 1. The trusted data curator uses the incomplete dataset D as input and aims to generate a synthetic data D^* of n rows with a mechanism $M : \mathcal{D} \to \mathcal{D}$ such that minimizes $d(f(\bar{D}) - f(D^*))$, where $f : \mathcal{D} \to \mathbb{R}^l$ is any utility metric function that the user is interested in, $d(\cdot, \cdot)$ is a distance metric and $M \circ M_\Phi$* offers $(\bar{\varepsilon}, \bar{\delta})$-DP to the ground truth data $\bar{D}$.

The problem mentioned above differs from our initial one only in the final aspect: instead of aiming for DP for the observed incomplete data, we target DP for the ground truth data. The missing mechanism limits the information available for synthetic data generation. Although initially appearing similar, ensuring privacy for incomplete data doesn't necessarily guarantee privacy for ground truth data. In Section 4.5, we delve into their relationship, showcasing scenarios where privacy for incomplete data may or may not extend to ground truth data. We also explore how the missing mechanism can serve as a sampling mechanism to enhance privacy and improve the utility of synthetic data.

4.4 Privacy for Incomplete Data

This section examines Problem 1 and explores methods for generating synthetic data from an incomplete private dataset. The section starts by discussing two vanilla methods that are found to be ineffective in the DP context and instead recommends adaptive recourse methods that are novel solutions that address both issues and produce better-quality synthetic data.

4.4.1 Vanilla Approaches

Complete row only and imputation first are two traditional methods for handling missing data. These methods either involve discarding rows with missing values or filling up missing information with values inferred from the observed data.

Complete Row Only Approach. This approach is effective when the missingness is completely at random (MCAR) since the distribution of each attribute remains the same after removing missing rows. However, for other types of missingness, such as missing at random (MAR) and missing not at random (MNAR), the complete row only approach can lead to biased results. Hence, a standard synthetic data generation algorithm that learns directly from the remaining complete rows will result in a biased data distribution that is different from the ground truth data.

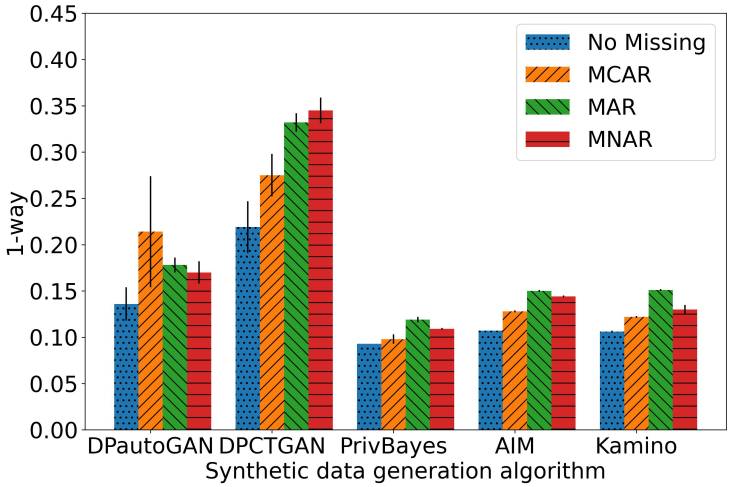


Figure 4.1: Complete row only approach results in poor results for MAR and MNAR missing mechanism. The vertical lines denote the std dev. across 3 runs.

Figure 4.1 illustrates the performance of five different DP synthetic data generation algorithms using the complete row only approach for the Adult dataset with various missing mechanisms. The results indicate that the complete row only approach performs poorly for missing at random (MAR) and missing not at random (MNAR) mechanisms, which introduce bias to the estimated distribution of attributes. This bias can directly affect statistical approaches such as PrivBayes which rely heavily on empirical estimation of marginals. The 1-way distance between the generated synthetic dataset and the original dataset confirms this observation.

Besides the potential bias issue for the complete row only approach, the number of complete rows remaining can be very small. For synthetic data generation methods involving large deep learning models such as GAN, feeding the training process with a small number of complete rows will result in a poor data generation model. This is because the learning process does not converge or/and the noise added for achieving DP overshadows the signals of the training samples. For example, the ground truth Adult dataset which has 32k rows reduces to $\approx$ 5k complete rows with 20% MAR and $\approx$ 1k complete rows with 20% MCAR/MNAR missing mechanism respectively. Our results show that the number of complete rows plays a vital role in the performance of the synthetic data generation algorithms. We discuss this in detail in Section 4.6 where we study several prior work approaches and evaluate them on the different missing data scenarios.

Imputation First Approach. Imputation is vastly used in practice where the missing data are filled up with values inferred from the observed data.

There are multiple ways to impute missing values in the dataset, including:

1. Statistical methods: Each attribute of the dataset is modeled separately using statistical methods such as mean, median, and mode. The model is then used to fill up the missing values of the attribute [107, 155]. For example, in Figure 4.2, for the left tables, we use the median of the observed values of column A, to fill up the missing value of the 4th row.

2. Hot and cold deck methods: This imputation technique replaces every missing value with another value from the same dataset (Hot deck) or from a proxy dataset (Cold deck) [13, 133]. The missing cells of the incomplete row are then filled up from the closest similar row. Similarity metric like cosine distance or ℓ_2 distance can be used.

3. ML imputation: Machine learning (ML) based approaches are common for missing data imputation [84, 102]. An ML model is trained to predict the missing values of an attribute based on other non-missing attributes in the dataset as training features.

In our private dataset setting, imputations must also be conducted privately. A straightforward yet ineffective method involves randomly selecting values from the missing attribute's domain, as illustrated later (Figure 4.5). We skip analysis of the cold deck imputation as finding another similar dataset is impractical for private datasets.

DP imputation can be approached in two manners. The first involves splitting the privacy budget, allocating a portion for imputation and the remainder for synthetic data generation. However, this approach is challenging due to the additional tuning of budget allocation as well as the choice of imputation algorithm. Additionally, some imputation techniques such as the hot deck imputation that are row specific (replicates the missing value in a row based on some other observed value of a different user) cannot be performed in the DP setting. Randomizing this row to achieve DP introduces too many errors to the dataset.

The second way is to formulate imputation as a transformation of the dataset and calculate the associated privacy cost as an end-to-end algorithm. We use the notion of stability (Theorem 9) to calculate the privacy costs of these transformations.

Definition 9. *[120] We say a transformation $T(\cdot)$ is c-stable, if the distance between $T(D)$ and $T(D')$ is at most c times the distance between D and D'. The composite mechanism $\mathcal{M} \circ T$ then becomes $(c \cdot \varepsilon, \delta)$-DP, for any mechanism $\mathcal{M}$ which is (ε, δ)-DP.*

Lemma 5. *Consider a transformation $T_A(\cdot)$ for imputing attribute A, which takes in the incomplete dataset D as part of the input and outputs a dataset D' with the complete values for attribute A. Then, the stability of $T_A(\cdot)$ is $c = m_A + 1$ where m_A refers to the number of missing values for the attribute A.*

Proof. As the neighboring databases D and D' differ by a row, $T_A(\cdot)$ uses two different values x and x' to impute the missing values in D and D' respectively. As there are $m_{A(1)}$ rows in both D and D' that have missing values for attribute A, the resulted imputed databases, $T(D)$ and $T(D')$ have $m_{A(1)} + 1$ number of rows (include the row that D and D' differ). □

Using the above Lemma we can see that applying a sequence of imputation functions over the attributes of a dataset $(\cdots, T_{A_i}, \cdots\}$, the difference in the resulted datasets can be very large when the input dataset differs in a single row. Note that these results hold even if imputation functions for two attributes are not the same.

Theorem 7. *The composite mechanism $\mathcal{M} \circ T$ on a dataset D with n rows is $n\varepsilon$-DP, where $\mathcal{M}$ is a ε-DP mechanism, and T is a sequence of imputation functions performed to each attribute of D.*

Proof. Consider a worst-case scenario: D and D' differ in a single row that does not have any missing values, and the rest of the rows have only one attribute with missing values, i.e., $\sum_i m_{A_i} = n - 1$. As T uses the complete row to impute all the missing values, all rows will be affected and the overall cost of $M \circ T$ will be $n\varepsilon$-DP. □

Example 8. *In Figure 4.2, we illustrate two worst-case toy examples for a dataset with two columns using mean and median imputations. The gray color indicates missing values and dotted lines denote differing rows between the top and the bottom datasets for each example. The missing values in both columns in the left and right examples are filled up with mean and median functions respectively. After applying imputations on the neighboring datasets D and D', the number of rows in column B in both examples starts to differ by 4 rows and 5 respectively including the imputed values and the differing row. In such a scenario, one needs to pay 4ε or 5ε privacy cost to ensure DP to the incomplete data.*

4.4.2 Adaptive Recourse Approach

Both aforementioned approaches suffer from inefficiencies in data or privacy budget utilization. The complete row approach discards partial rows, wasting potential learning data, while the imputation-first method incurs high privacy costs. These challenges prompt us to modify existing synthetic data generation techniques to optimize both dataset information and privacy budget usage, which we term *adaptive recourse*. The concept involves employing the privacy-preserving learning aspect of synthetic data generation for both imputation and synthetic data sampling. This offers two advantages: the privacy budget is dedicated

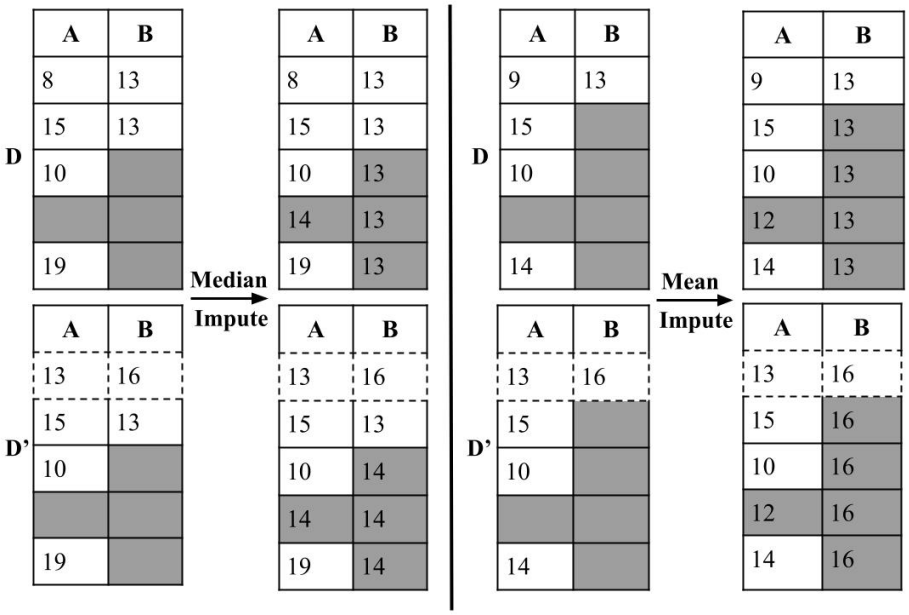


Figure 4.2: Illustration of worst case statistical imputations.

solely to learning a single model, and the imputation process generates more comprehensive training examples, enhancing model utility. We have selected three representative DP data generation models—generative adversarial networks (GAN), partial marginal observation, and column-wise data generation—as proof-of-concept for these adaptive approaches, which can extend to other existing or new DP models.

GAN-based adaptive recourse. In non-private literature, several approaches use the GAN framework to deal with missing data [190, 104, 115, 189]. We choose to privatize an approach called MisGAN [104], which allows us to simultaneously demonstrate the power of learning the data distribution and the missingness pattern for GAN-based algorithms. We call its DP version, DP-MisGAN, as shown in Algorithm 6. We first describe the high-level architecture of MisGAN and DP-MisGAN, and then we highlight the enhancement in this approach.

Algorithm Overview: MisGAN/DP-MisGAN trains two generator-discriminator pairs — one for learning the data distribution and the other for learning the missingness pattern. The training spans E epochs, with each epoch sampling $|D|/B$ sized subsets from the dataset D without replacement (Line 4). Each subset S_t is processed with real data x_{data} (Line 6) and its corresponding missing mask x_{mask}(Line 7). The missing mask x_{mask}, computed from the missing indicator matrix M, marks where data is missing as 1 and 0 otherwise. Missing values in real data are replaced with 0s (Line 8). Two fake examples y_{data} and y_{mask} are generated by passing random Gaussian noise through data and mask

generators (Line 9). These generators are updated using gradient descent, with discriminators learning true distributions in one phase (Line 10) and generators updating within specified T_G intervals in the second phase (Line 11). In each generator interval, two fake samples y_{data} and y_{mask} are again generated similarly from the two generators (Line 12) and gradients from the discriminator is computed (Line 13). Note that the non-private MisGAN uses these gradients directly to update the parameters of the two generators (Line 16) and releases both generators in the end. Finally, after the training is completed for E epochs, the discriminators are thrown away, and the privately learned data generator is used to sample synthetic data (Line 20) .

Highlights: In our DP-MisGAN, we clip and then noise the gradients learned for the generators using the *sampled Gaussian mechanism* (SGM) [123] to ensure privacy (Lines 14-15). This step is different from prior efforts for DPCTGAN [59] and DPautoGAN [166], which privatize non-private optimizer of the discriminators using DPSGD [181, 20, 162, 6]. This divergence in technique leads to notable difference in the algorithm's utility: while the discriminators become noisy due to the introduction of noise, the generator's gradient calculation, which relies on discriminator weights (Line 13), is also affected. We observe that we only need to publish the generators, not the discriminators. Hence, we resort to the gradient sanitization (GS) approach [34] to perturb only the gradients of the generators (Lines 14-15), without affecting the utility of the discriminator. In addition, the GS approach also allows us to skip the hyperparameter tuning for the gradient clip C, which can be vastly detrimental if set wrong [126]. Unlike the standard gradient clipping to bound the sensitivity of the gradient norm by C, i.e., $g/max(1, \|g\|_2/C)$, we clip the gradient by $g/max(1, \|g\|_2)$ (Line 15) by considering a Wasserstein-Gan (WGAN) [14] framework with an additional gradient penalty term in the loss function of the discriminator that enforces the ℓ_2-norm of the discriminator gradients to be naturally close to 1.

Algorithm 6: DP-MISGAN

Require: Incomplete dataset D, noise scale σ, epochs E, learning rates η_D and η_G, generator interval T_G, batch size B, missing indicator matrix M

Initialize data generator θ_G^D and discriminator θ_D^D
Initialize mask generator θ_G^M and discriminator θ_D^M
for i in $[1, \ldots, E]$ **do**
  Subsample dataset D into $\{S_k\}_{k=1}^{k=(|D|/B)}$ subsets
  for t in $[1, \ldots, |D|/B]$ **do**
    Set real data $x_{data} = S_t$
    Sample real mask x_{mask} from missing indicator $M(S_t)$
    Fill missing values in x_{data} with 0
    Generate fake data and mask
    $y_{data} = \theta_G^D(z)$ where $z \sim \mathcal{N}(0, 1)$
    $y_{mask} = \theta_G^M(z)$ where $z \sim \mathcal{N}(0, 1)$
    Update $\theta_D^D = \theta_D^D - \eta_D \nabla_{\theta_D^D}[\frac{1}{B}(\theta_D^D; x_{data}) - \frac{1}{B}(\theta_D^D; y_{data})]$
    $\theta_D^M = \theta_D^M - \eta_D \nabla_{\theta_D^M}[\frac{1}{B}(\theta_D^M; x_{mask}) - \frac{1}{B}(\theta_D^M; y_{mask})]$
    if t in interval of T_G **then**
      Generate fake data and mask
      $y_{data} = \theta_G^D(z)$ where $z \sim \mathcal{N}(0, 1)$
      $y_{mask} = \theta_G^M(z)$ where $z \sim \mathcal{N}(0, 1)$
      Compute data and mask gradient
      $g_d = \nabla_{\theta_G^D}(\theta_D^D; y_{data})$, $g_m = \nabla_{\theta_G^M}(\theta_D^M; y_{mask})$
      Clip each gradient $\bar{g}_{id} = g_{id} / \max(1, \|g_{id}\|_2)$
      $\bar{g}_{im} = g_{im} / \max(1, \|g_m\|_2)$
      Compute noisy gradients $\tilde{g}_{id} = \bar{g}_{id} + 2\sigma\mathcal{N}(0, 1)$
      $\tilde{g}_{im} = \bar{g}_{im} + 2\sigma\mathcal{N}(0, 1)$
      Update generators $\theta_G^D = \theta_G^D - \eta_D \frac{1}{B} \tilde{g}_{id}$
      $\theta_G^M = \theta_G^M - \eta_D \frac{1}{B} \tilde{g}_{id}$
    end if
  end for
end for
Generate and return synthetic dataset from generator θ_G^D

A tight bound for the privacy loss of the training procedure can be calculated using Rényi-DP (RDP).

Definition 10 (Rényi-DP [122])**.** *A randomized algorithm $M : \mathcal{D} \to \mathcal{R}$ is said to have ε-Rényi-DP of order α or (α, ε)-RDP for short if for any pair of neighbouring databases*

$D, D' \in \mathcal{D}$ that differ in one tuple, it holds that: $\frac{1}{\alpha-1}\log \mathbb{E}_{x\sim\mathcal{M}(D')}\left(\frac{P_D(x)}{P_{D'}(x)}\right)^{\alpha} \leq \varepsilon$, *where* P_D *and* $P_{D'}$ *are the output probability densities of* $\mathcal{M}(D)$ *and* $\mathcal{M}(D')$ *respectively.*

The post-processing and composability properties of DP also apply to RDP. Specifically, if a sequence of adaptive mechanisms $\mathcal{M}_1$, $\mathcal{M}_2$, $\cdots$, $\mathcal{M}_k$ satisfy (α, ε_1)-, (α, ε_2)-, $\cdots$, (α, ε_k)-RDP, then the composite privacy loss is $(\alpha, \sum_{i=1}^{k}\varepsilon_i)$-RDP. The RDP privacy loss of the *sampled Gaussian mechanism* (SGM) [123] is given using Lemma 6.

Lemma 6. *Given a database* D *and query* $f : \mathcal{D} \to \mathbb{R}^d$ *with sensitivity* S_f*, returning* $f(\{x \in D \mid x \text{ is sampled with probability } r\}) + \mathcal{N}(0, S_f^2\sigma^2\mathbb{I}^d)$ *results in the following RDP cost for an integer moment* α*.*

$$R_{\sigma,r}(\alpha) = \begin{cases} \frac{\alpha}{2(S_f\sigma)^2} & r = 1 \\ \sum_{k=0}^{\alpha}\binom{\alpha}{k}(1-r)^{\alpha-k}r^k\exp(\frac{\alpha^2-\alpha}{2(S_f\sigma)^2}) & 0 < r < 1 \end{cases}$$

We can now calculate the total RDP cost of DP-MisGAN by composing the cost of each SGM in the training procedure.

Theorem 8. *The total RDP cost of DP-MisGAN is*

$$R(\alpha) = 2\lceil\frac{T}{T_G}\rceil\sum_{k=0}^{\alpha}\binom{\alpha}{k}(1-\frac{B}{|D|})^{\alpha-k}\frac{B}{|D|}^{k}\exp(\frac{\alpha^2-\alpha}{8\sigma^2})$$

where T *is the total number of iterations* $(E * |D|/B))$ *and* T_G *is the generator iterations.*

Proof. Let the gradient clipping procedure of the discriminator in DP-MisGAN be $f = g/\max(1, \|g\|_2/C)$. The sensitivity S_f can be thus be derived by the reversed triangle inequality.

$$S_f = \max_{D,D'}\|f(D) - f(D')\| \leq 2C = 2$$

The last inequality follows as we set $C = 1$ in DP-MisGAN. Each SGM in DP-MisGAN (Algorithm 6) has sampling probability of $\frac{B}{|D|}$ and thus the RDP cost of each SGM can be derived using Lemma 6 as $\sum_{k=0}^{\alpha}\binom{\alpha}{k}(1-\frac{B}{|D|})^{\alpha-k}\frac{B}{|D|}^{k}\exp(\frac{\alpha^2-\alpha}{8\sigma^2})$. DP-MisGAN consists of two generators that are updated every generator interval T_G each. Therefore, using the composition property of RDP, we can compose the costs for both the discriminators in DP-MisGAN as $R(\alpha) = 2\lceil\frac{T}{T_G}\rceil\sum_{k=0}^{\alpha}\binom{\alpha}{k}(1-\frac{B}{|D|})^{\alpha-k}\frac{B}{|D|}^{k}\exp(\frac{\alpha^2-\alpha}{8\sigma^2})$. □

By the tail bound property of RDP [122], we can convert the RDP cost of DP-MisGAN to (ε, δ)-DP, where ε is computed by

$$\varepsilon(\delta) = \min_{\alpha} R(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}, \tag{4.1}$$

for a given δ. In practice, given the parameters $\varepsilon, \delta, B, T, T_G$, the σ is calculated using Theorem 8 and the order α is usually searched within a range of user input values [180].

We expect DP-MisGAN to perform better than the naive GAN approaches because it learns from both the complete rows as well as the incomplete rows of the dataset. Furthermore, as DP-MisGAN learns the missing data pattern of the incomplete dataset, we anticipate that it will capture more information in complex MAR and MNAR missing mechanisms.

Flexibility: The framework of DP-MisGAN can be used for any GAN method. The core idea of the change is to encompass two discriminator-generator pairs for learning missing data and synthetic data generation. However, to achieve better privacy accounting, it is important to use the GS approach with a WGAN structure and discard the discriminator after training.

Partial marginal observation-based adaptive recourse. This approach can be applied to algorithms that use low dimensional marginal queries [193, 118, 150, 185, 117, 95, 147, 103]. Instead of discarding all the partially missing rows, only the rows with missing cells in the queried attributes are removed. Such a strategy is most helpful when only a subset of attributes have missing data. For example, with MAR missing mechanism, partial marginal observation can be learned from all the non-missing columns.

Algorithm overview: We extend PrivBayes [193] using this strategy and call it PrivBayes enhanced or PrivBayesE in short (Algorithm 7). Both PrivBayes and PrivBayesE learn Bayesian networks of degree k to know the correlated columns. The network B is initialized by adding the first attribute in the attribute list $\mathcal{A}$ as a vertex to the graph. The attributes(vertices) that are discovered are stored in the list V (Lines 1-2). Next the algorithm loops over each attribute in $\mathcal{A}$ (Line 3) and generates $|V|$ choose k sets appended with every attribute seen so far $\mathcal{A} \setminus V$ and stores in a list Ω (Lines 4-5). The mutual information values for each pair of attributes in Ω is computed and the pair with highest value is chosen using exponential mechanism with privacy budget $\varepsilon_1/\mathcal{A}$ (Line 6). The edge connecting the two vertices with largest mutual information is then added to the network B (Line 7). PrivBayes/PrivBayesE then generate a synthetic dataset D^* using this network B in sequence (Line 10). Marginals are computed for each attribute X_i with its most correlated attributes Ψ_i using privacy budget $\varepsilon/2$ and added to D^* (Line 12-14).

Algorithm 7: PrivBayes Enhanced (PrivBayesE)

Require: Incomplete dataset D, Attributes $\mathcal{A}$, Privacy budget $\varepsilon_1, \varepsilon_2$

1: Initialize Bayesian network B of degree k and $V = \phi$
2: Sample X_1 from $\mathcal{A}$ and add (X_1, ϕ) to B; add X_1 to V
3: **for** $i = 2 \dots |\mathcal{A}|$ **do**
4: Initialize $\Omega = 0$
5: For each $X \in \mathcal{A} \setminus V$ and each $\Psi \in \binom{V}{k}$; add (X, Ψ) to Ω
6: Select a pair (X_i, Ψ_i) from Ω with maximal mutual information in complete rows for attributes X_i in D using exponential mechanism of budget $\varepsilon_1/|\mathcal{A}|$
7: Add (X_i, Ψ_i) to B; add X_i to V
8: **end for**
9: Initialize synthetic dataset D^*
10: **for** $i = 1 \dots |\mathcal{A}|$ **do**
11: Compute distribution from non-missing values $\Pr[X_i, \Psi_i]$ from the complete rows of X_i in D
12: Learn $\Pr^*[X_i, \Psi_i]$ with Laplace mechanism at budget ε_2
13: Set negative values to 0 and normalize
14: Sample from $\Pr^*[X_i, \Psi_i]$ and add to D^*
15: **end for**
16: Return D^*

Highlights: In PrivBayesE, modifications are made to the dataset generation process. Each time a marginal query is made, PrivBayesE learns from all non-missing information of the attribute(s) (Line 11). This improves upon the complete row approach by discarding missing rows on a smaller set of attributes, rather than from the entire dataset. This is particularly advantageous for scenarios like missing completely at random (MCAR), where analyzing more data aids in better estimating the true distribution of marginals, and missing at random (MAR), where some marginals are completely available, allowing estimation based on complete data. The privacy analysis of PrivBayesE mirrors that of PrivBayes, as PrivBayesE does not introduce additional queries to the dataset.

Flexibility: This enhancement can be applied as a wrapper to any method that makes use of low-dimensional marginals to generate synthetic data. In our paper, we choose PrivBayes as our baseline as it is the most simple and fundamental marginal based approach.

Column-wise data generation-based adaptive recourse. This approach can be applied to any algorithm that uses column-wise intermediate models to learn the data distribution. In such algorithms, a sequence of attributes is decided, and starting with the

second attribute in sequence, a model is learned to predict the current attribute using previously learned ones.

Algorithm overview: We extend Kamino [62] using this strategy and call in Kamino impute or KaminoI in short (Algorithm 8). Kamino/KaminoI starts with deciding a sequence of attributes based on a given denial constraints Ψ (Line 1). The distribution of the first attribute in the computed sequence is learnt using all the non-missing cells (Line 3). This computed distribution is noised (Line 4) and values are sampled to populate the synthetic dataset (Line 5). For each new attribute Y in the sequence, Kamino/KaminoI learns a private intermediate model which uses all previously visited attributes X to predict the new attribute Y (Lines 7-8). This intermediate model is used to generate the values for the attribute Y in the synthetic data given the sampled values for X (Line 10).

Highlights: KaminoI includes an additional imputation step to impute values of attribute Y (Line 9), utilizing the same intermediate model that was trained to predict attribute Y given attribute set X. This enhancement ensures that no missing values are discarded; instead, they are used to train intermediate models. It is worth noting that the sequence S significantly influences KaminoI's imputation process. In Kamino, the sequence is generated considering input constraints Ψ. However, if an attribute with many missing values is positioned early in sequence S, its imputation may be less effective. To optimize imputation, attributes not in constraints Ψ can be ordered based on decreasing percentage of missing values. If available, clues from the missing mechanism can also help determine the sequence. For instance, with the MAR mechanism predicting missing IQ based on age, the age attribute can precede the age column in S. To ensure fair comparison, the same sequence as Kamino is used for KaminoI in the experimental section.

Flexibility: This enhancement is applicable to any method that iterates over the columns of the dataset. Such an algorithmic architecture allows for value imputation as learning progresses. Each time a model predicts the next attribute, it can also impute missing values using the same model, incurring no additional privacy costs. This strategy is particularly effective when missing data correlates with other attributes in the dataset, such as in the missing at random (MAR) scenario. While there's only one known private approach employing this strategy [62], several non-private approaches exist [188, 154]. Although PrivBayesE can use learned distributions to impute missing values for all visited attributes (Line 12 in Algoirthm 7), it only utilizes low-way marginals compared to KaminoI's larger models, which benefit more from imputation. Hence, we did not include this imputation step in PrivBayesE.

Algorithm 8: Kamino Impute (KaminoI)

Require: Incomplete dataset D, Attributes $\mathcal{A}$, Constraints Ψ, Privacy budget $\varepsilon_1, \varepsilon_2$

Build sequence S of attributes $\mathcal{A}$ using constraints Ψ
Initialize synthetic dataset D^*
Compute distribution of first attribute $H = \Pr[S_1]$ using all non-missing values
Generate DP H^* by adding Gaussian noise of budget ε_1
Sample from H^* to populate $D^*[S_1]$
for $i = 2 \dots |\mathcal{A}|$ **do**
    Load training features $X = S_{:j}$, and target label $Y = S_j$
    Train model $M = \theta(X, Y)$ privately with budget $\frac{\varepsilon_2}{|\mathcal{A}|-1}$
    Impute missing values in dataset $D[S_j]$ using M
    Predict synthetic values $\Omega = M(D^*[S_{:j}])$ and fill $D^*[S_j] = \Omega$
end for
Return D^*

4.5 Privacy for Groundtruth Data

In this section, we shift our focus to exploring the privacy implications for the ground truth data, which we approach as a distinct problem that closely relates to Problem 1. We first find that the solutions for Problem 1 do not always offer sufficient privacy for Problem 2, except when the probability of missing values in a row is independent of the other rows in the dataset (Section 4.5.1). Furthermore, we demonstrate that certain missing mechanisms, such as MCAR, allow a tighter privacy analysis for Problem 2 when applying the same solution from Problem 1 in Section 4.5.2.

4.5.1 Relationship to Problem 1

We have proposed multiple synthetic data generation algorithms M which train on the incomplete dataset D and achieve (ε, δ)-DP as solutions to problem 1. However, this incomplete dataset D results from a missing mechanism M_ϕ on the ground truth dataset $\bar{D}$. In problem 2, we study the same mechanisms M which train on D but discuss their privacy impact on $\bar{D}$. We do so by combining the missing mechanism M_ϕ and the synthetic data generation process M as a composite mechanism $M \circ M_\phi$.

It is important to note that just because M is a DP mechanism for incomplete data D, it does not necessarily mean that $M \circ M_\phi$ is DP for the ground truth data $\bar{D}$.

Example 9. *In Figure 4.3, consider neighboring ground truth data $\bar{D}$ and $\bar{D}'$ differ in the*

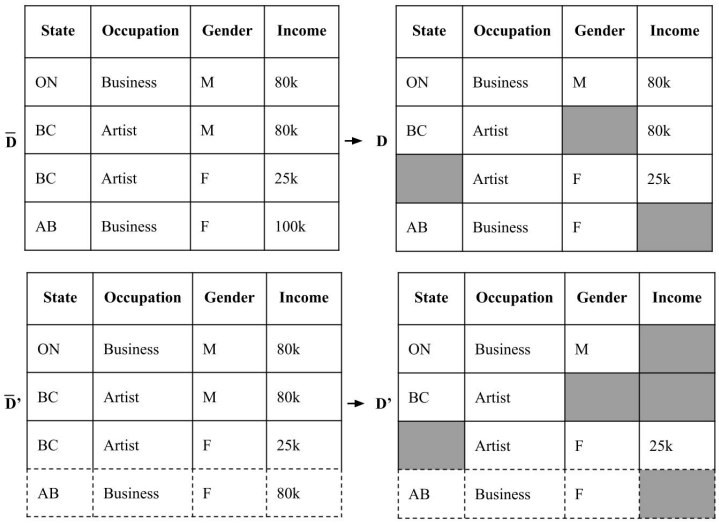


Figure 4.3: Example of private incomplete datasets generated from two neighbouring ground truth datasets. Gray denotes missing cells and dotted lines represent the differing row.

last row's income value (100k v.s. 80k). Their income columns have an MNAR missing mechanism that hides the highest income value and their corresponding incomplete data D and D' then differ more than one row. This means that an ε-DP mechanism for incomplete data cannot guarantee the same level of privacy for the ground truth data.

The example above does not provide a strong privacy guarantee for the ground truth data because the probability of a row having missing values depends on the values of other rows. However, we can show that if M_ϕ enforces independent probabilities for each row to have missing values, a strong privacy guarantee applies to the ground truth data.

Theorem 9. *Let the missing mechanism M_Φ have independent randomness to hide the values of each row and $D = M_\phi(\bar{D})$. If M achieves (ε, δ)-DP for D, then $M \circ M_\Phi$ satisfies $(\bar{\varepsilon}, \bar{\delta})$-DP for $\bar{D}$, where $\bar{\varepsilon} \leq \varepsilon, \bar{\delta} \leq \delta$.*

Proof. As M_Φ has independent randomness to hide values of each row, given the ground truth data $\bar{D}$ and a possible incomplete dataset D, we have $\Pr[D|\bar{D}] = \prod_l \Pr[D_l|\bar{D}]$, where D_l refers to the value taken by the lth row. Consider neighboring groundtruth datasets $\bar{D}$ and $\bar{D}'$ differ in the ith row and any possible output O of $M \circ M_\Phi$. Let $\mathcal{D}$ be all possible incomplete datasets that can be outputted by M_Φ from $\bar{D}$ or $\bar{D}'$. We partition $\mathcal{D}$ into $\{\cdots, \mathcal{D}_j, \cdots\}$ such that all datasets with the same row values except the ith row are in the same group $\mathcal{D}_j$. Hence, for all $D \in \mathcal{D}_j$, they have the same probability for $\prod_{l \neq i} \Pr_{M_\Phi}[D_l|\bar{D}]$. Now we have

$$
\begin{aligned}
& \Pr[O|\bar{D}] \\
= & \sum_{\mathcal{D}_j} \sum_{D \in \mathcal{D}_j} \Pr_{M}[O|D] \Pr_{M_\Phi}[D|\bar{D}] \\
= & \sum_{\mathcal{D}_j} \sum_{D \in \mathcal{D}_j} (\Pr_{M}[O|D] \Pr_{M_\Phi}[D_i|\bar{D}] \cdot \prod_{l \neq i} \Pr_{M_\Phi}[D_l|\bar{D}]) \\
= & \sum_{\mathcal{D}_j} (\prod_{l \neq i} \Pr_{M_\Phi}[D_l|\bar{D}] \cdot \sum_{D \in \mathcal{D}_j} \Pr_{M}[O|D] \Pr_{M_\Phi}[D_i|\bar{D}]) \\
\leq & \sum_{\mathcal{D}_j} (\prod_{l \neq i} \Pr_{M_\Phi}[D_l|\bar{D}'] \cdot \sum_{D' \in \mathcal{D}_j} (e^{\varepsilon} \Pr_{M}[O|D'] + \delta) \Pr_{M_\Phi}[D'_i|\bar{D}']) \\
= & e^{\varepsilon} \sum_{\mathcal{D}_j} \sum_{D' \in \mathcal{D}_j} \Pr_{M}[O|D'] \Pr_{M_\Phi}[D'|\bar{D}] + \delta \sum_{\mathcal{D}_j} \sum_{D' \in \mathcal{D}_j} \Pr_{M_\Phi}[D'|\bar{D}'] \\
= & e^{\varepsilon} \sum_{\mathcal{D}_j} \sum_{D' \in \mathcal{D}_j} \Pr_{M}[O|D'] \Pr_{M_\Phi}[D'|\bar{D}] + \delta
\end{aligned}
$$

The inequality above is based on for any neighbors D and D', we have $\Pr_M[O|D] \leq e^{\varepsilon} \Pr_M[O|D'] + \delta$ and $\sum_{D \in \mathcal{D}_j} \Pr_{M_\Phi}[D_i|\bar{D}] = \sum_{D' \in \mathcal{D}_j} \Pr_{M_\Phi}[D'_i|\bar{D}'] = 1$. □

Theorem 9 says that the privacy bound for the ground truth dataset is lesser than equal to the bound of the incomplete dataset for a synthetic data generation algorithm if each row in the ground truth dataset has an independent probability of having missing values. Next we illustrate how to obtain a tighter privacy bound for the missing completely at random (MCAR) mechanism.

4.5.2 Privacy Amplification Due To MCAR

Missing completely at random (MCAR) enforces an independent probability of having missing rows for each attribute in the dataset. We use these probabilities to tighten the privacy bounds for ground truth data when the missing mechanism is MCAR. The technique we developed is inspired by the seminal work of privacy amplification due to sampling [17]. The premise of privacy amplification by subsampling is that we run a DP algorithm on some random subset of the data (e.g., sampled Gaussian mechanism, DP-SGD). The subset introduces additional uncertainty, which benefits privacy. Privacy amplification due to subsampling has been shown to work for many sampling methods (e.g., Poisson sampling,

sampling with/without replacement) and for neighboring datasets which may differ with replacement or substitution. Privacy amplification by subsampling theorem 10 makes this intuition precise.

Theorem 10 (Sampling Amplification Theorem [17, 163])**.** *Consider an algorithm $M : \mathcal{D} \to \mathcal{D}$ that satisfies (ε, δ)-DP and a sampling mechanism $S(D)$ that samples a random subset U from dataset D of n samples. If $p = max_{i \in [n]} \Pr_U[i \in U]$, then the composite mechanism $M(S(D))$ offers (ε', δ')-DP where $\varepsilon' = \log(1 + p(e^{\varepsilon} - 1))$, $\delta' = p\delta$. For small values of ε, we have $\varepsilon' = \log(1 + p(e^{\varepsilon} - 1)) \approx p\varepsilon$.*

In our missing data context, we note that for synthetic data generation algorithms that train on incomplete data, many rows are naturally discarded due to the presence of missing cells. We exploit this natural throwing out of rows as a sampling mechanism and show that it can be used to amplify privacy. Recall from Section 4.2 that MCAR enforces independent probability of having missing cells in the dataset for each attribute $\phi_1, \cdots, \phi_k$. We use these probabilities to propose our amplification results in Proposition 4.

Proposition 4. *Consider an MCAR mechanism $M_\Phi : \mathcal{D} \to \mathcal{D}$ with missing probabilities $\{\phi_1, \ldots, \phi_k\}$ over attributes $\{A_1, \ldots A_k\}$ of the input ground truth data $\bar{D}$ and outputs an incomplete dataset D. If a (ε, δ)-DP algorithm $M : \mathcal{D} \to \mathcal{D}$ takes in rows in D which have no missing values on attributes $\mathcal{A}_M \subseteq \{A_1, \ldots, A_k\}$, then $M \circ M_\Phi$ offers $(p\varepsilon, p\delta)$-DP to the ground truth data $\bar{D}$ where $p_{\mathcal{A}_M} = \prod_{A_i \in \mathcal{A}_M}(1 - \phi_i)$. We call $\mathcal{A}_M$ an amplification attribute set for M and $p_{\mathcal{A}_M}$ the amplification factor of $\mathcal{A}_M$.*

Proof. A row in MCAR has $\prod_{i=1}^{i=l}(1 - \phi_i)$ probability of having no missing values and plugging in to Theorem 10. □

We note three important facts. First, if an algorithm M takes in rows with no missing values over an attribute set $\mathcal{A}_M$, then M also takes in rows with no missing values over an attribute set $\mathcal{A}'_M \subset \mathcal{A}_M$. In other words, if $\mathcal{A}_M$ is an amplification attribute set for M, then any subset of $\mathcal{A}_M$ is an amplification attribute set for M with amplification factor greater than that of $\mathcal{A}_M$. Second, when $\mathcal{A}_M = \emptyset$, $p_{\mathcal{A}_M} = 1$. Third and more importantly, as the dataset is read only once, each attribute can only be used once as an amplification factor. We can now use Proposition 4 and Theorem 10 in conjunction to show the privacy amplifications for the different algorithms we have discussed so far in our paper.

Use case 1: Privacy amplification for complete row only approach. Here we show how to apply Proposition 4 to all complete row only approaches (PrivBayes, Kamino, and GAN-based approaches). As these approaches take as input all attributes, the probability of seeing a row without missing values is $\prod_{i=1}^{i=k}(1 - \phi_i)$. The following example illustrates

how this probability can be used to obtain a tighter privacy bound for the ground truth data.

Example 10 (MCAR amplification for complete row only approach)**.** *Consider the incomplete dataset from Figure 4.3. Let's assume that the missing data comes from an MCAR mechanism where the missing probabilities are* $\phi_{State} = \frac{1}{4}, \phi_{Occupation} = 0, \phi_{Gender} = \frac{1}{4}, \phi_{Income} = \frac{1}{4}$ *. Given 4 DP sub-algorithms* M_1, M_2, M_3, M_4 *that each offer DP guarantee to the incomplete dataset* D *at budget* $\frac{\varepsilon}{4}$*.* M_1 *computes the marginals of the complete rows over attribute <State>,* M_2 *over <Occupation>,* M_3 *over <Gender> and* M_4 *over <Gender, Income>. As all sub-algorithms take as input only the complete rows, using Proposition 4, the amplification is* $\prod_i (1 - \phi_i) = 0.421$ *and using Theorem 11 the final privacy is* $\bar{\varepsilon} = 4 * 0.421 \frac{\varepsilon}{4} = 0.421\varepsilon$*.*

Use case 2: Privacy amplification for partial marginal observation approach. For partial marginal observation methods (e.g. PrivBayesE), calculating the amplification privacy cost is more complex. These methods involve multiple low-dimensional marginals with overlapping attributes. To determine the overall amplification for such algorithms, it is necessary to calculate the amplification for each marginal and carefully compose them. The complexity of this calculation arises from the optimal selection of the amplification attribute set for each marginal, which maximizes amplification while ensuring that each attribute is used only once. First, we consider a simple case that the amplification factors of all marginals are disjoint. In this scenario, we can compose the total privacy cost using Theorem 11 and demonstrate using Example 11.

Theorem 11. *Consider an MCAR mechanism* M_Φ*, and a sequence of* j *mechanisms* $M_1, \ldots, M_j$ *with DP guarantees of* $\varepsilon_1, \ldots, \varepsilon_j$ *to* D *and amplification attribute set* $\mathcal{A}_{M_1}, \ldots, \mathcal{A}_{M_j}$ *respectively. If their amplification attribute sets do not overlap, then these mechanisms offers DP to the ground truth data* $\bar{D}$ *at a cost of* $\bar{\varepsilon} = \sum_{i=1} p_{\mathcal{A}_{M_i}} \varepsilon_i$*.*

Proof. As all mechanisms M_i work on disjoint sets of attributes, their amplification attribute sets $\mathcal{A}_i$ are also disjoint. Furthermore as the missing probabilities are always ≤ 1, we always use all attributes in $\mathcal{A}_i$ amplify marginal M_i. We can then use Theorem 10 to calculate the final amplified privacy cost $\bar{\varepsilon} = \sum_{i=1} p_{\mathcal{A}_{M_i}} \varepsilon_i$. □

Example 11. *Continuing from Example 10, assume we have the same dataset but use a partial observation algorithm. We consider only the sub-algorithms* M_1, M_2*, and* M_4 *for this example. The marginals for these sub-algorithms do not overlap and allow us to consider all engaging attributes as their amplification attribute set. Hence, by Theorem 11,* M_1 *is amplified using* $p_{M_1} = 1 - \phi_{state} = \frac{3}{4}$*,* M_2 *is amplified using* $p_{M_2} = 1 - \phi_{occupation} = 1$ *and*

M_4 using $p_{M_4} = (1 - \phi_{gender})(1 - \phi_{income}) = \frac{9}{16}$. The amplified privacy cost would thus be $\frac{3}{4}\frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{9}{16}\frac{\varepsilon}{3} = 0.77\varepsilon$.

The problem however gets more nuanced when two marginals have overlapping attributes. We show this in Example 12 by first showing a naïve composition and then an optimized one.

Example 12. *Consider all 4 sub-algorithms in Example 10 and a partial observation algorithm. The marginals for sub-algorithms M_3 and M_4 overlap in the 'Gender' attribute with amplification factors $p_{M_3} = (1 - \phi_{Gender}) = \frac{3}{4}$ and $p_{M_4} = (1 - \phi_{Gender})(1 - \phi_{Income}) = \frac{9}{16}$ respectively. We cannot apply Theorem 11 on M_3 and M_4's entire attribute set as the corresponding amplification attribute sets would then overlap on the 'Gender' attribute. A naïve solution would be to amplify the DP mechanism with the most amplification and skip the others. In our example, we would amplify only M_4 with amplification of $\frac{9}{16}$ and skip amplification for M_3. The total amplified privacy cost would thus be $\bar{\varepsilon} = \frac{3}{4}\frac{\varepsilon}{4} + \frac{\varepsilon}{4} + \frac{\varepsilon}{4} + \frac{9}{16}\frac{\varepsilon}{4} = 0.83\varepsilon$. However, a better bound can be calculated if overlapping mechanisms were grouped together and amplified using the intersecting attribute. For instance, both M_3 and M_4 can be amplified by an amplification factor of $\frac{3}{4}$ using the amplification attribute set 'Gender', resulting in a total privacy loss of $\frac{3}{4}\frac{\varepsilon}{4} + \frac{\varepsilon}{4} + \frac{3}{4}(\frac{\varepsilon}{4} + \frac{\varepsilon}{4}) = 0.81\varepsilon$ i.e. tighter than 0.83ε.*

In a more general setting, solving this problem requires us to make groups of the mechanisms with overlapping attributes and make sure that each group is amplified using distinct amplification factors.

Problem 3. *Consider an MCAR mechanism M_Φ and a sequence of j mechanisms $M_1, \ldots, M_j$ with DP guarantees of $\varepsilon_1, \ldots, \varepsilon_j$ to D, where M_i computes a marginal over attributes $\mathcal{A}_i$. We would like to find amplification attribute sets $\{\mathcal{A}_{M_1} \subseteq \mathcal{A}_1, \ldots, \mathcal{A}_{M_j} \subseteq \mathcal{A}_k\}$ and their corresponding amplification factor $p_1, \ldots, p_j$ for $M_1, ...M_j$, that gives the smallest DP cost to the ground truth data $\bar{D}$.*

One way to solve the above problem is by creating valid partitions of marginals and assigning each group in the partition an amplification attribute set such that all groups have disjoint attribute sets and all marginals from the same group are amplified using their own amplification attribute set.

Definition 11 (Valid partition)**.** *Given DP mechanisms $M_1, \ldots, M_j$ for computing marginals over attribute sets $\mathcal{A}_1, \ldots, \mathcal{A}_j$, a partition of these mechanisms $P = \{G_1, \ldots G_i\}$ is considered valid if it satisfies these conditions: (1) All mechanisms in the same partition are amplified with the same set of amplification attribute set and with the same amplification factors; and (2) The amplification attribute sets of all partitions are disjoint.*

The privacy cost for a valid partition is $\bar{\varepsilon} = \sum_{G_l \in P} p_{\mathcal{A}_{G_l}} \sum_{M_j \in G_l} \varepsilon_j$*, where* $\mathcal{A}_{G_l}$ *is the amplification attribute set for the mechanisms grouped into* G_l *and* $p_{\mathcal{A}_{G_l}}$ *is the corresponding amplification factor.*

A valid partition ensures each group's amplification attribute set is disjoint, ensuring each attribute is considered only once. To solve Problem 3, we select the partition with the least privacy cost that is also valid. Thus, all DP sub-mechanisms are amplified using the best disjoint attribute set. However, enumerating all valid partitions is intractable due to the exponential number of possibilities[1]. If the cardinality of $\mathcal{A}$ and the number of DP mechanisms j are small, then one can enumerate all possible solutions and choose the best one. However, for datasets with a large number of attributes, we show an initial pruning method to trim away bad solutions using Lemma 7.

Lemma 7. *A valid partition* P *solution to Problem 3 should have a non-empty amplification attribute set for all group* $G \in P$*.*

Proof. Suppose there are a sequence of DP sub-mechanisms $M_1, \ldots, M_j$ with privacy cost of $\varepsilon_1, \ldots, \varepsilon_j$. A valid partition P over $M_1, \ldots, M_j$ consists of a group G that has an empty amplification attribute set. Then the amplification factor of group G will be $p_G = 1$ and the overall privacy cost of this partition can be calculated using Definition 11.

$$\begin{aligned}
\bar{\varepsilon}_P &= \sum_{G_l \in P} p_{\mathcal{A}_{G_l}} \sum_{M_j \in G_l} \varepsilon_j \\
&= \sum_{G_l \in \{P-G\}} p_{\mathcal{A}_{G_l}} \sum_{\mathcal{A}_j \in G_l} \varepsilon_j + \sum_{M_j \in G} \varepsilon_j \\
&= L + X
\end{aligned}$$

where $L = \sum_{G_l \in \{P-G\}} p_{\mathcal{A}_{G_l}} \sum_{\mathcal{A}_j \in G_l} \varepsilon_j$ and $X = \sum_{M_j \in G} \varepsilon_j$. We can always define another valid partition P' that has the groups as P but group G has amplification attribute set A. Therefore, the privacy cost of partition P' can be calculated as:

$$\begin{aligned}
\bar{\varepsilon}_{P'} &= \sum_{G_l \in P'} p_{\mathcal{A}_{G_l}} \sum_{M_j \in G_l} \varepsilon_j \\
&= \sum_{G_l \in \{P'-G\}} p_{\mathcal{A}_{G_l}} \sum_{\mathcal{A}_j \in G_l} \varepsilon_j + p_A \sum_{M_j \in G} \varepsilon_j \\
&= L' + p_A X
\end{aligned}$$

[1]The total number of partitions of a set is given by the Bell number.

, where $L' = \sum_{G_l \in \{P' - G\}} p_{\mathcal{A}_{G_l}} \sum_{\mathcal{A}_j \in G_l} \varepsilon_j$. Note that as amplification attribute set A is used for G in P', the privacy cost of the other groups can at most be increased by p_A. Assuming the worst case, $L' = p_A L$ and that the privacy cost of P is always higher than that of P'.

$$\begin{aligned} L + X - L' - p_A X &\geq 0 \\ L + X - p_A L - p_A X &\geq 0 \\ L(1 - p_A) + X(1 - p_A) &\geq 0 \\ (L + X)(1 - p_A) &\geq 0 \end{aligned}$$

The above inequality if always true as the first term is always positive (L and X are privacy costs) and the $p_A \leq 1$ as all missing percentages are ≤ 1. □

For large datasets where we are left with multiple partitions even after pruning, we use a brute force search as described in Algorithm 9. In Line 1, we enumerate all possible disjoint amplification attribute sets that we can make from $\mathcal{A}$ and store in a variable $\mathcal{P}_{\mathcal{A}}$. Then, we loop through each possible disjoint attribute set $P_{\mathcal{A}} \in \mathcal{P}_{\mathcal{A}}$, and calculate the cost of each amplification attribute set $\mathcal{A}_l \in P_{\mathcal{A}}$ in Line 2-3 using Proposition 4. We initialize the cost of the partition $c_{P_{\mathcal{A}}} = 0$ in Line 4. Then we loop through each pair of marginal S_i and its corresponding DP mechanism (M_i, ε_i) and find the candidate amplification attribute sets that are contained in the marginal S_i in Line 6. If no such candidate set is valid for S_i, then we can prune the entire partition $P_{\mathcal{A}}$ using Lemma 7 and loop back to Line 2. Otherwise, in Line 7-8, we find the candidate attribute set which has the best amplification cost for M_i and add its corresponding cost the final cost $c_{P_{\mathcal{A}}}$. Finally, the partition with the minimum sum cost is returned in Line 10. In Example 13 we show how Algorithm 9 can be used to find the valid partition for our running example with the lowest privacy cost.

Example 13. *Consider the same setup in Example 12. There are total of 4 attributes and Algorithm 9 starts by enumerating all 15 possible disjoint amplification attribute sets — {State | Occupation | Gender | Income}, {State Occupation | Gender | Income}, ... , {State Occupation Gender | Income}, ... , {State Occupation Gender Income}. We then iterate through each of these disjoint sets. Let's consider the disjoint sets* $\mathcal{A}_1 : \{State\}, \mathcal{A}_2 : \{Occupation\}, \mathcal{A}_3 : \{Gender\}, \mathcal{A}_4 : \{Income\}$. *For each attribute set we calculate its corresponding amplification factor,* $c_{\mathcal{A}_1} : (1 - \phi_{State}) = 3/4, c_{\mathcal{A}_2} : (1 - \phi_{Occupation}) = 0, c_{\mathcal{A}_3} : (1 - \phi_{Gender}) = 3/4, c_{\mathcal{A}_4} : (1 - \phi_{Income}) = 3/4$. *We then iterate through all marginals and choose the best amplification attribute for each marginal. Therefore,* M_1 *is amplified using* $\mathcal{A}_1$, M_2 *using* $\mathcal{A}_2$, M_3 *and* M_4 *both using* $\mathcal{A}_3$. *The final privacy cost therefore is* $\bar{\varepsilon} = \frac{3}{4}\frac{\varepsilon}{4} + \frac{\varepsilon}{4} + \frac{3}{4}\frac{\varepsilon}{4} + \frac{3}{4}\frac{\varepsilon}{4} = 0.81\varepsilon$. *This partition also happens to be the best partition among the 15 partitions.*

Algorithm 9: Optimal amplified privacy cost

Require: Marginals $\mathcal{S} = \{S_1, \ldots, S_j\}$ over attributes $\mathcal{A}$, DP mechanisms $\mathcal{M} = \{(M_1, \varepsilon_1), \ldots, (M_j, \varepsilon_j)\}$ for $\mathcal{S}$, Missing probabilities $\Phi = \{\Phi_A | A \in \mathcal{A}\}$ for MCAR

1: Find all possible partitions of $\mathcal{A}$ and store in $\mathcal{P}_\mathcal{A}$
2: **for** each attribute partition $P_\mathcal{A}$ in $\mathcal{P}_\mathcal{A}$ **do**
3: Calculate amplification factor $c_{\mathcal{A}_l} = \prod_{A \in \mathcal{A}_l}(1 - \phi_A)$ for $\mathcal{A}_l \in P_\mathcal{A}$
4: Initialize disjoint set cost $c_{P_\mathcal{A}} = 0$
5: **for** each $S_i \in \mathcal{S}$ with its corresponding $((M_i, \varepsilon_i))$ **do**
6: Skip $P_\mathcal{A}$ if $\{\mathcal{A}_l \subseteq S_i | \mathcal{A}_l \in P_\mathcal{A}\} = \emptyset$
7: Find the best amplification attribute set for M_i: $\mathcal{A}_{l*} \leftarrow argmin_{\mathcal{A}_l \in P_\mathcal{A} \wedge \mathcal{A}_l \subseteq S_i} c_{\mathcal{A}_l}$
8: Add the best amplified cost $c_{P_\mathcal{A}} = c_{P_\mathcal{A}} + \varepsilon_i \cdot c_{\mathcal{A}_{l*}}$
9: **end for**
10: **end for**
11: Return the attribute partition $P_\mathcal{A}$ with minimum cost $c_{P_\mathcal{A}}$

Use case 3: Privacy amplification for column-wise imputation algorithms. Column-wise imputation algorithms (e.g., KaminoI) learn attributes sequentially in a predefined sequence S over $A_1, \ldots, A_k$. The first attribute S_1 is learned using its observed distribution, while the rest $S_2, \ldots, S_k$ are learned using intermediate models $M_2, \ldots, M_k$ with privacy costs $\varepsilon_2, \ldots, \varepsilon_k$. At each i^{th} iteration, attribute S_i is learned using intermediate model M_i, taking previously learned attributes $S_{:i}$ as feature input. After training, M_i is used to sample the synthetic dataset and impute incomplete values of $D[S_i]$. Thus, for the next model M_{i+1}, previously learned attributes are either complete or imputed. The total number of complete rows fed to an intermediate model M_i depends solely on the complete values in $D[S_i]$, leading to amplified privacy for ground truth data $\bar{\varepsilon_i} = p_{S_i}\varepsilon_i$, where $p_{S_i} = 1 - \phi_i$. However, we note that for training model M_i, every attribute $S_{:i-1}$ is considered for amplification using the same probability as S_i. Since each attribute can be amplified only once, privacy for ground truth is calculated accordingly.

Theorem 12. *Consider an MCAR mechanism M_Φ and a sequence of attributes S and k mechanisms $M_1, \ldots, M_k$ with privacy cost of $\varepsilon_1, \ldots, \varepsilon_k$ to D, where M_i is an intermediate model that trains A_i as target and $A_{:i}$ as features. If model M_i is used for imputation of attribute A_i, then the overall process offers DP to the ground truth data $\bar{D}$ at a cost of $\bar{\varepsilon} = p_{S_j}\varepsilon_j + \sum_{i=1, i\neq j}^{k} \varepsilon_i$, where $j = \max_{i=1}^{k}(1 - \phi_i)$.*

Proof. As all cells of an attribute are imputed at previous iterations, the number of com-

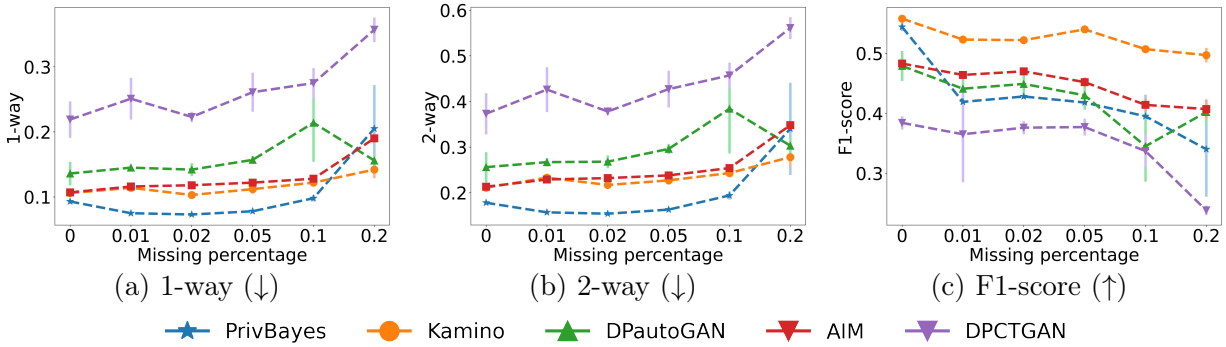
plete rows fed to a model at the ith iteration depends on the missing cells of the S_i attribute. The amplification for the ith model can therefore be calculated at $\varepsilon_i p_{S_i}$, where $p_{S_i} = 1 - \phi_i$ is the amplification factor of attribute S_i. To calculate the overall privacy of the algorithm, we would like to compose the amplified privacy of all sub models. However, the model M_i trains on every attribute $S_{:i-1}$ and each attribute uses the same probability as S_i. Furthermore as each attribute can only be used once, we can only amplify a single model. Hence, we choose to amplify the model with the maximum amplification M_j where $j = \max_{i=1}^{k}(1 - \phi_i)$ and the other models are left un-amplified. □

Example 14. *Consider the incomplete dataset from Figure 4.3. Lets assume we have a column based imputation algorithm learns this dataset. Each attribute is learnt using the equal privacy budget* $\frac{\varepsilon}{4}$*. The attributes 'State', 'Gender' and 'Income' has an amplification factor* $\frac{3}{4}$ *whereas the attribute 'State' has a factor of* 1*. Therefore, the amplified privacy budget is* $\bar{\varepsilon} = \frac{3}{4}\frac{\varepsilon}{4} + 3\frac{\varepsilon}{4} = 0.9375\varepsilon$.

Discussion on MAR and MNAR. MAR and MNAR mechanisms model probabilities of missing data conditioned upon non-missing values from other columns (MAR) or missing values from the same column in the dataset (MNAR). Unfortunately, verifying the presence of MAR and MNAR missing types in the dataset lacks a foolproof method, as it may rely on observed and unobserved variables and their interactions (as discussed in Section 4.2). Due to the conditional nature of these missing types, achieving privacy amplification with MAR or MNAR mechanisms is challenging. Each row's probability of having missing values may vary due to conditional dependence on dataset values. While maximum probabilities or noisy upper bounds can be used for privacy amplification if these probabilities are known or privately learned, they are often associated with multiple patterns and complex relationships. For instance, in the simple example depicted in Figure 4.3, rows with the highest income value (80k) are missing. Here, the probability of a row having a missing value depends directly on the probability of a row having an income of 80k. To calculate missing probabilities associated with MNAR, not only the highest value in the column but also its probability of occurrence needs to be determined. Without distribution assumptions, such calculations cannot be achieved with DP [29]. Amplification extensions for MAR and MNAR remain a topic for future research.

4.6 Evaluation

We thoroughly experiment with DP synthetic data generation algorithms on missing data. First, we demonstrate how existing DP methods are affected by varying amount of missing



(a) 1-way (↓) (b) 2-way (↓) (c) F1-score (↑)

Figure 4.4: The effect of missing data on DP synthetic data generation algorithms.

Table 4.1: Dataset Characteristics

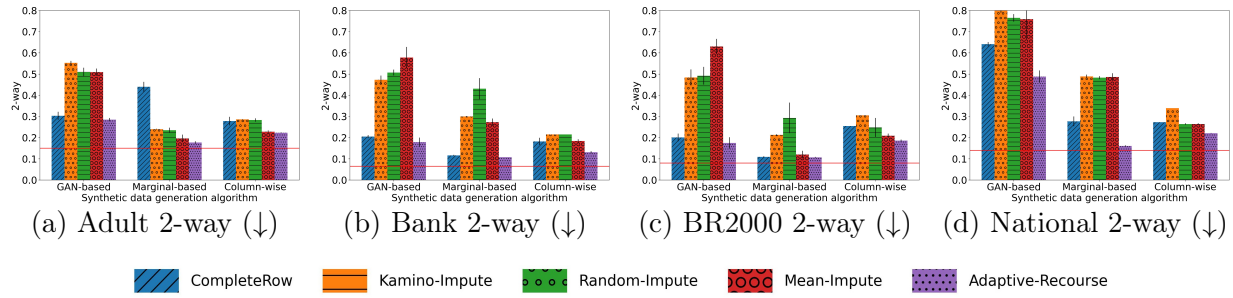
Dataset	Cardinality	#Numerical Attr	#Categorical Attr
Adult	32561	5	10
Bank	45211	3	14
BR2000	38000	3	11
National	15012	6	14

data. Next, we evaluate the effectiveness of our proposed adaptive recourse methods and analyze the impact of varying missing data percentages, missing mechanisms, and privacy budgets for each method. Finally, we show how missingness amplifies privacy for ground truth data.

4.6.1 Experimental Setup

Datasets.

We run our experiments on four tabular datasets as described in Table 4.1: (i) Adult dataset [48], which contains information about 32561 individuals from the 1994 US Census (ii) Bank dataset [129], which has 45211 rows about direct marketing campaigns of a Portuguese banking institution (iii) BR2000 [193], which consists of 38,000 census records collected from Brazil in the year 2000, (iv) National dataset [169] from NIST Diverse Community Excerpts which contains information about 15012 individuals US census data. Each dataset has a combination of numerical and categorical columns which are pre-processed according to the synthetic data generation algorithm as discussed in their respective research paper – numerical attributes are discretized into 10 uniform bins or scaled between 0 to 1 and the categorical attributes are encoded using one-hot or ordinal encoding.

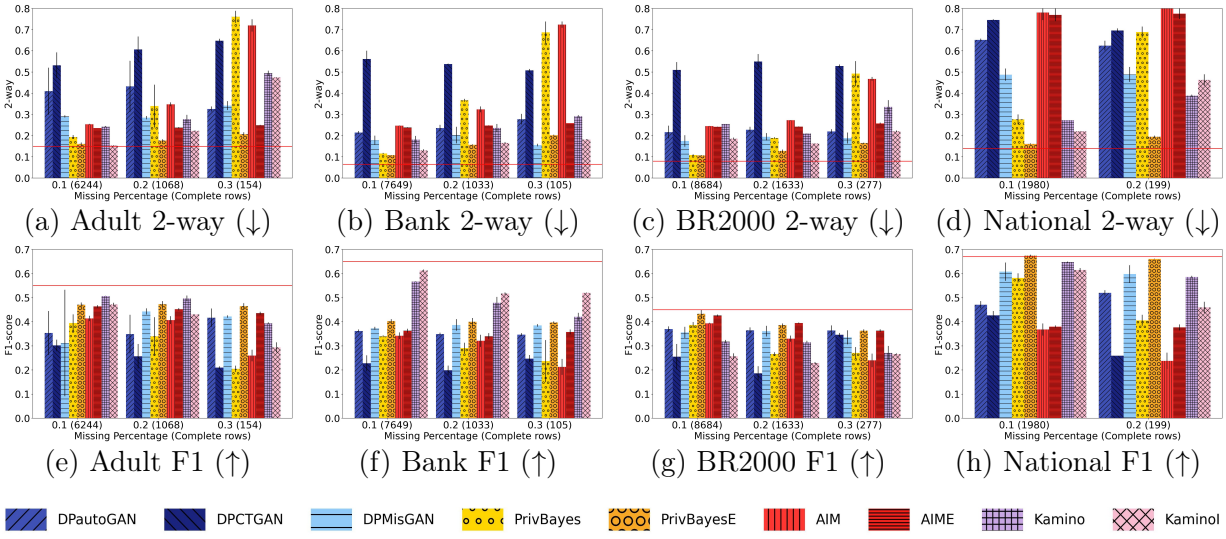


(a) Adult 2-way (↓) (b) Bank 2-way (↓) (c) BR2000 2-way (↓) (d) National 2-way (↓)

Figure 4.5: Comparing all strategies to deal with missing data. Adaptive recourse results in the best performance, followed by the complete row approach. The red line represents the best no-missing baseline.

Baselines. We consider several existing differentially private data generation methods that do not consider missing data: PrivBayes [193] and AIM [117] from statistical approaches, DPCTGAN [59] and DPautoGAN [166] from deep learning techniques, and Kamino [62], which is a mixed approach. These methods have been published in well-known conferences with their code readily available. For each method, we construct the following baselines to deal with missing data as discussed in Section 4.4.1. The first baseline referred by its original name combines the original method with the complete row-only approach. Next, we construct baselines for the imputation first approach, which first imputes the data and then generates the synthetic data. We initially considered differentially private data imputation using k-means [40] and OLS regression [41], but they fail in working with categorical columns and require a large privacy budget for the imputation process itself. Hence, we adopt the following imputation methods: (i) random imputation, which fills missing values randomly from the attribute domain; (ii) mean imputation, replacing missing numerical attribute values with the mean and sampling from the probability distribution for categorical attributes; and (iii) Kamino imputation, which leverages intermediate models from Kamino for imputation and shares similar model as Holoclean [183, 3]. Random impute requires no privacy budget, but the other two require splitting of the privacy budget for imputation and for the data generation algorithm.

Parameters. The utility of the algorithms replies upon multiple hyperparameters, and we try our best to tune these parameters by running grid searches. Although we don't consider privacy costs for tuning, it's crucial to tune these parameters privately in practice [126, 139]. Deep learning methods typically require a larger privacy budget for meaningful results. Hence, we assign PrivBayes and Kamino $\varepsilon = 1$, and GAN approaches $\varepsilon = 3$ by default. Adaptive methods maintain the same budget as their non-adaptive counterparts. PrivBayes operates under pure DP ($\delta = 0$), while others employ approximate DP. The δ



(a) Adult 2-way (↓) (b) Bank 2-way (↓) (c) BR2000 2-way (↓) (d) National 2-way (↓)

(e) Adult F1 (↑) (f) Bank F1 (↑) (g) BR2000 F1 (↑) (h) National F1 (↑)

Figure 4.6: Adaptive methods (DPMisGAN, PrivBayesE, KaminoI) vs their respective baselines using the complete row approach (DPautoGAN, PrivBayes, Kamino) with MCAR missingness at $\varepsilon = 1$. Algorithms of the same category are colored with the same shade. The adaptive methods result in better quality synthetic data. The red line denotes the best no missing baseline.

value is approximately set one magnitude lower than $1/|D|$ to the nearest exponent of 10.

Missing data. We implement a pipeline that can generate different categories of missing data at different missing percentages, using the approach from Muzellec et al. [130]. MCAR missing data is generated by masking the original dataset using a realization of a Bernoulli variable with a fixed parameter such that there is exactly the required number of missing values. To generate MAR data, we first use 50% of the attributes in the dataset as features for a logistic regression model. The other attributes then have missing values according to random weights in the logistic model. A bias term is fitted using line search to attain the desired proportion of missing values. The MNAR approach works similarly to MAR, with the difference that the 50% non-missing attributes are masked by an MCAR mechanism. This imposes the logistic model to depend potentially on missing values, hence enforcing MNAR missingness. As 50% of the columns in the MAR mechanism has complete values, it comparatively has more complete rows in comparison to MCAR and MNAR. We run experiments on all kinds of missing types for every dataset and go up to 30% missing values except the national dataset, where we stop at 20% due to the lower number of rows in the original dataset.

Metrics.

We use two utility metrics for evaluation. The first metric is k-way marginal distance. We calculate all k-sized combinations of attributes in the dataset [193, 62], and then report the average variational distance between the true vs the synthetic marginals. For each k-sized combination set of attributes in $A \in \mathcal{A}$, we calculate the marginal, $h : \mathcal{D} \rightarrow \mathbb{R}^{|\mathcal{D}(A)|}$ and report the average variational distance between as $\max_{a \in \mathcal{D}(A)} |h(D')[a] - h(\bar{D})[a]|$ where D' and $\bar{D}$ are the synthetic data and ground true data respectively [193, 62]. In our evaluation, we set $k = 1, 2$. Smaller values of this metric show more closeness between the true and synthetic data. Our second metric is model training. We consider 9 classification models (LogisticRegression, AdaBoost, GradientBoost, XGBoost, RandomForest, BernoulliNB, DecisionTree, Bagging, and MLP) to classify each attribute in the dataset (e.g., income is more than 50k or not, the loan should be given or not to user) using all other attributes as features. Each target attribute is processed to be a binary attribute, and we try to balance the two classes as much as possible. The quality of the learning task is represented by the average of all models across all attributes. The F1 score is reported for learning quality. Each model is trained using 70% of the synthetic data, and the F1 score is evaluated using 30% of the true data [62]. Larger values of this metric show better performance. For all our experiments, we run 3 times and report the mean with standard deviation.

4.6.2 Experimental Findings

We evaluate the impact of missing values on DP synthetic data generation algorithms with different missing mechanisms and varying missing data from 1% to 30%. The quality of synthetic data generated by all algorithms degrades drastically, particularly when having $> 10\%$ of missing data.

Benchmark Existing DP Methods on Missing Data

We demonstrate the impact of missing values on DP synthetic data generation algorithms. Figure 4.4 shows the performance of the baselines on the Adult dataset with varying levels of missing completely at random (MCAR) data (x-axis). Our results indicate that missing data up to 5% minimally affects the quality of the synthetic data with 1-way and 2-way metrics experiencing 5-19% impact, and F1-score experiencing 1-13% impact. However, as missing data increases, the quality of synthetic data generated by all algorithms degrades drastically. With 20% missing data, the 1-way metric is affected by 14-190%, the 2-way by 18-147%, and the F1-score is impacted by 6-28%. With high amounts of missing data ($\geq 10\%$), PrivBayes and DPCTGAN exhibit the most decrease in utility. However, Kamino and DPAutoGAN show a consistent level of performance even with more missing

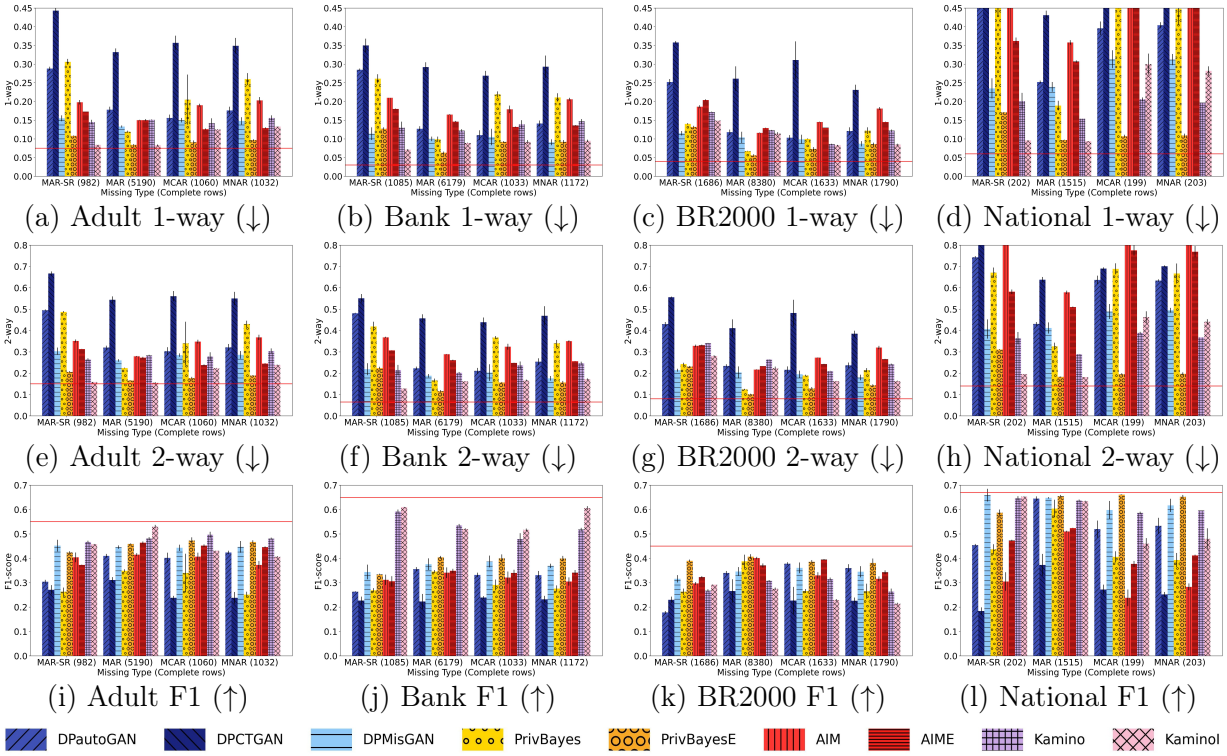


Figure 4.7: Adaptive methods vs non-adaptive methods on different missing mechanisms. The red line denotes the no missing baseline.

data indicating a high degree of stability in the output. The stability of these methods can be attributed to the fact that these methods tend to extract more information from the available data. Kamino learns the functional dependencies of the dataset and tries to preserve them while generating the synthetic data and DPAutoGAN trains an autoencoder as a preprocessing step to learn the low-dimensional statistics of the data.

Evaluate different strategies to deal with missing data

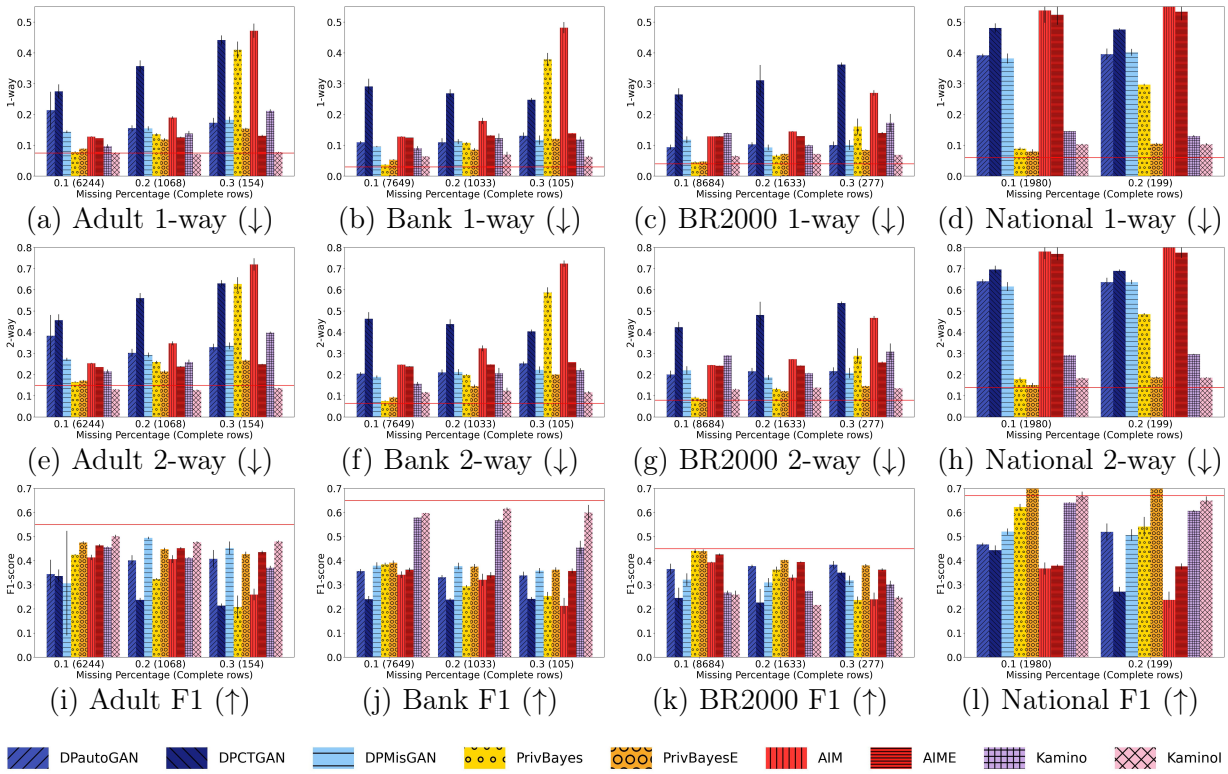
For each existing DP algorithm for generating synthetic data, we compare its respective baseline methods and its adaptive recourse approach. Figure 4.5 shows the performance of these methods across all four datasets, employing 20% MCAR missing values. The red line denotes the baseline approach's performance without missing values. For mean and Kamino imputation, we allocate three splits of the privacy budget (25%, 50%, and 75%) for imputation and plot the average and standard deviation across all splits, reserving the rest for generating the synthetic dataset. We observe that no single imputation strategy

consistently outperforms others, with the adaptive recourse approach generally yielding the best results. Hence, allowing learning and imputation to happen together is crucial. Additionally, the complete row approach often performs second best, serving as our baseline for subsequent experiments.

Evaluate Adaptive Recourse Approach

We evaluate adaptive recourse approaches at various experimental configurations.

Varying missing percentage. In this experiment, we compare the best baseline from the previous experiment that uses the complete row approach with its corresponding adaptive recourse approach from Section 4.4.2. We repeat our experiment on four different datasets with varying amount of MCAR data at two privacy levels ($\varepsilon = 1, 3$).



(a) Adult 1-way (↓) (b) Bank 1-way (↓) (c) BR2000 1-way (↓) (d) National 1-way (↓)

(e) Adult 2-way (↓) (f) Bank 2-way (↓) (g) BR2000 2-way (↓) (h) National 2-way (↓)

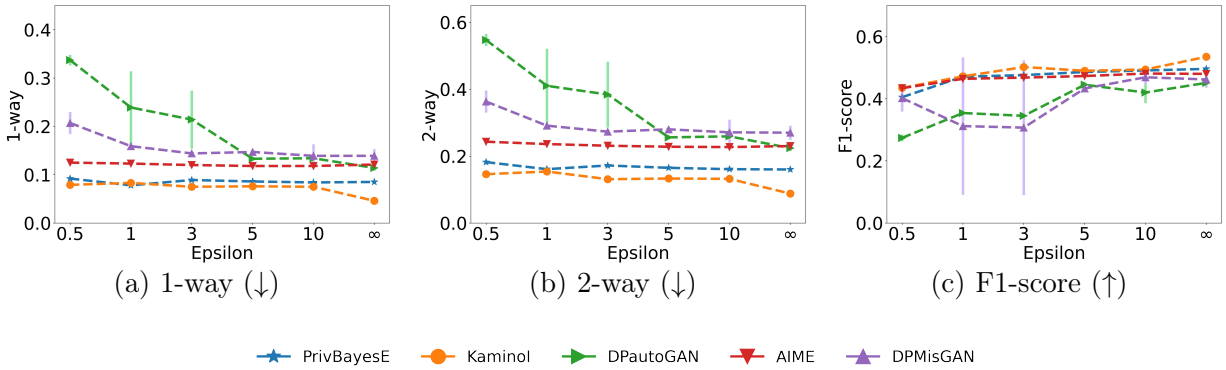
(i) Adult F1 (↑) (j) Bank F1 (↑) (k) BR2000 F1 (↑) (l) National F1 (↑)

Figure 4.8: Baselines vs Adaptive methods at $\varepsilon = 3$. The adaptive methods perform better than their respective baselines.

In Figure 4.6, we plot the algorithms in different shades of color depending on their

type. We observe in general that adaptive recourse approaches (DPMisGAN, PrivBayesE, AIME, and KaminoI) result in significantly better quality synthetic data compared to their baseline with the complete row only approach (DPautoGAN, DPCTGAN, PrivBayes, AIM and Kamino). Across all datasets, the 1-way scores are improved by up to 68%, 2-way by up to 66% and F1-scores of up to 24%. Furthermore, we observe that the adaptive methods often achieve the same utility as the no missing baseline with 10% missing data (e.g., top left subfigure Adult 2-way for KaminoI and PrivBayesE at 10% missing data).

Varying missing mechanisms. In Figure 4.7, we repeat our experiment for all 4 datasets at 10% missing data for all missing mechanisms. The adaptive recouse approaches(DPMisGAN, PrivBayesE, AIME and KaminoI) beat their non-adaptive complete-row baselines across all missing mechanisms. As missing values are added only to half of the attributes, missing at random (MAR) has more complete rows (the number in brackets on the x-axis labels) as compared to the other mechanisms. It is interesting to note that if we increase the missing percentage for MAR and plot it with same rows (MAR-SR), the algorithms start performing poorly. Hence, we make the conclusion that the number of complete of rows makes a more vital impact compared to the missing mechanism itself.



(a) 1-way (↓) (b) 2-way (↓) (c) F1-score (↑)

Figure 4.9: Varying privacy budget and comparing utilities of our approaches.

Varying privacy budget. We show the impact of the privacy budget in Figure 4.9 for adaptive methods with 10% MCAR missing data on Adult. We vary the privacy budget $\varepsilon \in [0.5, 1, 3, 5, 10, \infty]$ where $\varepsilon = \infty$ refers to the non-private run. First, we note that increasing the privacy budget improves the utility of the synthetic dataset across all algorithms. Second, we observe that PrivBayesE and KaminoI outperformed GAN-based approaches at all privacy budgets. Similar observations are found in Figure 5.9 and Figure 4.7. The poor performance of the GAN-based approaches is primarily due to the requirement for specific preprocessing steps and extensive hyperparameter tuning, which was challenging. Third, we find that there is no clear winner for all datasets, all utility

Table 4.2: Amplified privacy for ground truth data.

Dataset	MCAR missing %				
	0.1	0.2	0.3	0.4	0.5
Adult	0.88	0.77	0.65	0.47	0.44
BR2000	0.83	0.68	0.55	0.41	0.31

metrics, and all privacy budgets, which is consistent with the observations from prior benchmarks on DP synthetic data generation [168]. However, we observe that PrivBayesE performs better at smaller epsilon values ($\varepsilon \leq 1$), but KaminoI has better performance with higher epsilon ($\varepsilon > 1$). We attribute this phenomenon to the fact that KaminoI trains multiple intermediate models, and these models generally require a higher privacy budget. Therefore, we recommend PrivBayesE at a low privacy regime ($\varepsilon \leq 1$) and KaminoI at a higher privacy regime ($\varepsilon > 1$).

In addition, we observe that PrivBayesE outperforms others at the 2-way tasks (Figure 5.9 top row), and KaminoI demonstrates superior performance in the F1-score metric even at a low privacy regime ($\varepsilon = 1$, Figure 5.9 bottom row, except BR2000). This shows that depending on the data sets and the downstream tasks different methods perform best. Hence, we also recommend to an end user to test all methods for their specific data sets and downstream tasks and assess result quality in the way we have done. Then they can select the best method for their case, but this ignores privacy cost of algorithm selection. Generally, we recommend marginal-based approach like PrivBayesE for k-way tasks and KaminoI for ML tasks.

Amplification Due To Missingness

In this experiment, we show the amplified privacy budget for ground truth data. We experiment with PrivBayesE that runs on the incomplete dataset with 10 - 50% MCAR missing data. For each run, we allocate a privacy budget of $\varepsilon = 1$ and observe the marginals calculated by PrivBayesE. We assume a uniform privacy budget for each marginal calculated by PrivbayesE and run Algorithm 9 to calculate the best valid partition of these marginals. In Table 4.2, we plot the amplified privacy cost $\bar{\varepsilon}$ based on the best partition found by the algorithm. We repeat the experiment for two datasets – Adult and BR2000. Our results show that the amplified privacy cost decreases almost linearly from 0.88x to 0.44x for Adult and 0.83x to 0.31x for the BR2000 dataset.

Chapter 5

Hyperparameter Tuning: Honest Tuning using Adaptive Optimizers

5.1 Introduction

In this chapter, we change gears and look into the hyperparameter tuning problem of an end-to-end DP system. Specifically, we look into tuning DP optimizers for training private machine learning models. DP training procedures that have been developed [181, 20, 162, 5] generally resemble non-private gradient-based methods, but with the incorporation of gradient clipping and noise injection. These steps add additional hyperparameters on top of the non-private parameters that need to be tuned for optimum performance of the models.

Hyperparameter selection is instrumental to achieving high accuracy. The most common methods are grid search or random search, both of which incur a computational overhead scaling with the number of hyperparameters under consideration. In the private setting, this issue is often magnified as most private training procedures introduce new hyperparameters. Regardless, and more importantly, hyperparameter tuning on a sensitive dataset also costs in terms of *privacy*, naively incurring a multiplicative cost which scales as the square root of the number of candidates (based on composition properties of differential privacy [88]).

Most prior works on private learning choose not to account for this cost [5, 192, 176], focusing instead on demonstrating the accuracy achievable by private learning under idealized

conditions; if the best hyperparameters were somehow known ahead of time. Some works assume the presence of supplementary public data resembling the sensitive dataset [16, 152], which may be freely used for hyperparameter tuning. Naturally, such public data may be scarce or nonexistent in settings where privacy is a concern, leaving practitioners with little guidance on how to choose hyperparameters in practice. As explored in our paper, poor hyperparameter selection with standard private optimizers can have catastrophic effects on model accuracy.

Hope is afforded by the success of adaptive optimizers in the non-private setting. The canonical example is Adam [94], which exploits moments of the gradients to adaptively and dynamically determine the learning rate. It works out of the box in many cases, providing accuracy comparable with tuned SGD. However, Adam has been overlooked in the context of private learning since previous works have show than fine-tuned DPSGD tends to perform better than DPAdam [138], which has lead to several subsequent works [191, 176] to limit themselves to highlighting accuracy under ideal DPSGD hyperparameters. We navigate the different options available to a practitioner to solve the *honest private hyperparameter tuning problem* and ask, *are there optimizers which provide strong privacy, require minimal hyperparameter tuning, and perform competitively with tuned counterparts?*

5.2 Preliminaries for DP Optimizers

In this section, we provide the required preliminaries for this chapter based on specific types of DP optimizers, delineating the various hyperparameters required for each.

DP-SGD: The most popular private optimizer is the differentially private stochastic gradient descent (DPSGD) [181, 20, 162, 5]. DPSGD takes individual steps for each point in the sampled lot just like in SGD. Due to these individual steps, SGD is more locally unstable and empirically generalizes better than other optimizers [198]. However, SGD requires the learning rate to be properly tuned when changing architectures or datasets, without which SGD may show subpar performance. There are five main hyperparameters involved in DPSGD. We start with those also present in the non-private setting, highlighting any differences that arise due to privacy.

- Training iterations (T) - In the private setting, more iterations results in a larger privacy cost.
- Lot size (L) - Lot size factors into the privacy calculation, due to amplification by subsampling [18].

- Learning rate (α) - Learning rate has an important interplay with the clipping threshold C, discussed in Section 5.5.

The following hyperparameters are new in the private setting.

- Clipping threshold (C) - To limit sensitivity, per-example gradients are clipped to have ℓ_2-norm bounded by C.
- Noise scale (σ) - Scale of the noise added, as a multiple of C. A larger value gives higher privacy but (typically) lower accuracy.

DPMomentum: The private counterpart of SGD-Momentum [156, 151], which adds the momentum parameter to the update rule of DPSGD [74]. This optimizer adds an extra hyperparameter to tune as no default value for momentum is known.

DP-Adam: Adam [94] is an adaptive optimizer that combines the advantages from AdaGrad [49] and RMSProp [77]. At the core of Adam, exponentially averaged first and second moment estimates of the gradients are used to take a step. Converting Adam to its differentially private counterpart DPAdam can be done trivially by replacing the standard gradients with their clipped and noised counterparts. Adam adds two extra hyperparameters (β_1, β_2) to tune in the DP setting. However, default values of these parameters are known in the non-private setting. We will tune these parameters to the private setting in Section 5.5. The adaptivity of these optimizers imply they need not be tuned across learning rates, hence reducing a hyperparameter to tune.

ADADP: This DP adaptive optimizer finds the best learning rate at every alternate iteration [96]. It does so by leveraging the ℓ_2 error of taking a full step and taking two half steps. If the error computed is greater than a threshold τ, the learning rate is updated using a closed form expression. As suggested by the authors, for all our experiments using ADADP, we use the threshold $\tau = \sqrt{\frac{d}{2T}}$, where d is the model dimension and T is the total number of iterations.

5.3 Problem Setup and Overview

Consider a sensitive dataset D which lies beyond a privacy firewall and has n points of the form $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $x_i \in \mathcal{X}$ is the feature vector of the ith point and $y_i \in \mathcal{Y}$ is its desired output. Though our experiments are carried out in the supervised

setting, all results can be translated to unsupervised setting as well. The dataset has been divided into two parts, the training set and the validation set. A trusted curator wants to train a machine learning model by making queries on the dataset with a total end-to-end training privacy budget of $(\varepsilon_f, \delta_f)$ such that the model can perform with high accuracy on the validation set. The curator wants to try multiple hyperparameter candidates for the model to figure out which candidate gives the maximum accuracy. However, as the model is private, each candidate requires multiple queries made on the dataset and all of them need to be accounted in the total privacy budget of $(\varepsilon_f, \delta_f)$.

Note that any validation accuracy must also be measured privately. Since this accuracy is a low-sensitivity query with a scalar output, and must only be computed once per choice of hyperparameters, the cost of this procedure is generally a lower order term versus the main training procedure. Thus for simplicity, we do not noise these validation accuracy queries. As we will see later, some optimizers require more candidates to tune and hence would also require more privacy budget than others.

To tackle private hyperparameter selection, we first compare the available private tuning procedures in Section 5.4. We show that the privacy cost for training a model depends on the hyperparameter grid size and standard composition theorems provide the best guarantees when the grid is small. In Section 5.5, we investigate different optimizers to see how many candidates are required to output a good solution. In Section 5.5, we provide theoretical and empirical evidence to demonstrate an intrinsic coupling between two hyperparameters – the learning rate and clipping norm in DPSGD. We show that this coupling makes DPSGD sensitive to these parameter choices, which can drastically affect the validation accuracy. In Section 5.5, we demonstrate that an adaptive optimizer, DPAdam, translates well from the non-private setting and obviates tuning of the learning rate. In Section 5.6, we empirically compare DPAdam with DPSGD and DPMomentum to show that DPAdam performs at par with less hyperparameter tuning. Finally, in Section 5.7, we establish that DPAdam converges to a static learning rate in restricted settings, and unveil a new optimizer DPAdamWOSM which can leverage this converged value without computing the second moments.

5.4 Privately Tuning DP Optimizers

Effective hyperparameter tuning is crucial in extracting good utility from an optimizer. Unlike the non-private setting, DP optimizers typically i) have more hyperparameters to tune; ii) require additional privacy budget for tuning. Existing work on DP optimizers acknowledge this problem (e.g., [5]), but do not address the privacy cost incurred during

hyperparameter tuning [5, 192, 176]. There are two main prior general-purpose approaches for private hyperparameter selection. The first performs composition via Moments Accountant [5], and the second is the algorithm of [109] (LT). The latter is a theoretical result, and to the best of our knowledge, has not been previously evaluated in practice. We investigate the privacy cost of these two techniques in practice and discuss situations in which each method is preferred.

Algorithm 10: LT algorithm for private selection of (ε, δ)-DP input algorithms

Require: $\gamma \leq 1$, $\delta_2 > 0$, and sampling access to candidates and their scores $Q(D)$

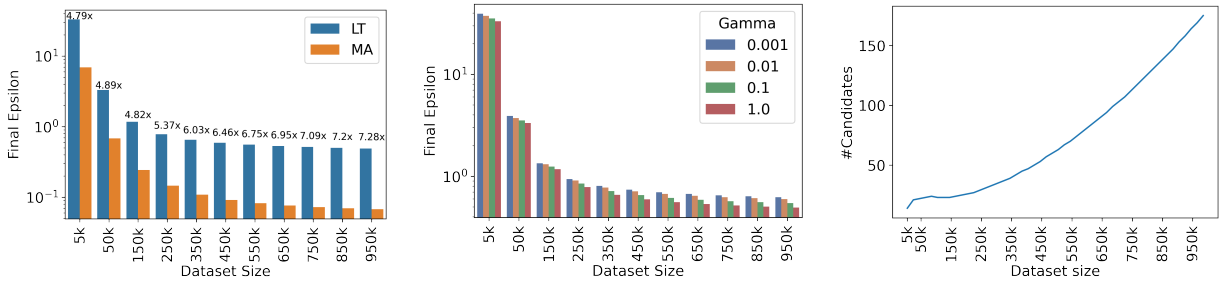
Initialize the list $S = \emptyset$
Initialize $\Upsilon = \frac{1}{\gamma} \log \frac{1}{\delta_2}$
for $j \in [1, \Upsilon]$ **do**
    Draw $(x, q) \sim Q(D)$
    $S \leftarrow S \cup (x, q)$
    Flip a γ-biased coin, output highest scored candidate from S and halt;
end for
Output highest scored candidate from S

Tuning cost via LT [109] propose a random stopping algorithm (LT) to output a 'good' hyperparameter candidate from a pool of K candidates, $\{x_1, \ldots, x_K\}$. They assume sampling access to a randomized mechanism $Q(D)$ which samples $i \sim [K]$, and returns the i-th candidate x_i, and a score q_i for this candidate. It is a random stopping algorithm, in which at every iteration, a candidate is picked from Q i.i.d. with replacement and a γ-biased coin is flipped to randomly stop the algorithm. When the algorithm stops, the candidate with the maximum score seen so far is outputted. In the approximate DP version of this algorithm, an extra parameter Υ is set to limit the total of number of iterations. The pseudocode of this algorithm is shown in Algorithm 10.

Theorem 13 ([109], Theorem 3.4)**.** *Fix any $\gamma \in [0, 1]$, $\delta_2 > 0$ and let $\Upsilon = \frac{1}{\gamma} \log \frac{1}{\delta_2}$. If Q is $(\varepsilon_1, \delta_1)$-DP, then the LT algorithm is $(\varepsilon_f, \delta_f)$-DP for $\varepsilon_f = 3\varepsilon_1 + 3\sqrt{2\delta_1}$ and $\delta_f = \sqrt{2\delta_1}\Upsilon + \delta_2$.*

Theorem 13 expresses the privacy cost of the algorithm in terms of the privacy cost of individual learners, and parameters of the algorithm itself. The δ_2 parameter does not significantly affect the final epsilon ε_f of the algorithm and in practice, one can set it to a very small value (10^{-20}). Though a small value of δ_2 has little effect on δ_f, it increases the hard stopping time of the algorithm, Υ.

To understand the LT algorithm, we will compare the privacy costs of training a single hyperparameter candidate with a final ε_f, δ_f budget via LT and compare it with the privacy cost ε_1, δ_1 of the underlying individual learner. This setting might seem unnatural for LT as it was designed to select from a pool of candidates but we choose this setting to show the minimum privacy cost overhead associated with LT and later show how the privacy cost changes for multiple candidates (varying γ). To use LT, one needs to figure out the ε_1, δ_1 via Theorem 13 using the final ε_f, δ_f values and in this case, $\gamma = 1$ (as we have just one candidate). The individual learner is then trained using ε_1, δ_1 budget.

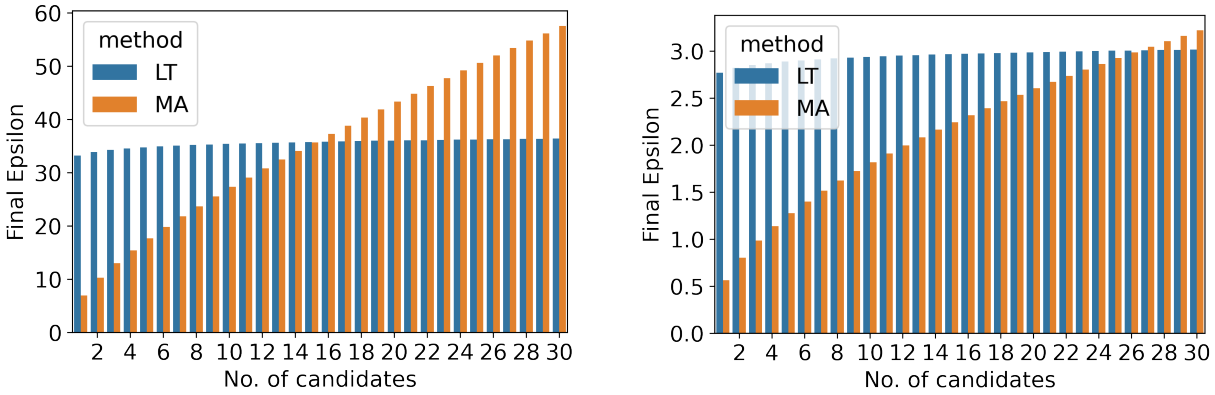


(a) (b) (c)

Figure 5.1: Comparing the privacy cost of LT versus Moments Accountant. The minimal privacy overhead incurred by LT is at least ≈5x, and increases with the dataset size (left). However, as we allow LT to sample and test more candidate hyperparameters, the privacy cost barely increases (middle). Moments Accountant is able to test a significant number of candidates at the same cost as the minimal privacy overhead of LT (right).

Due to the delicate balance of δ_f in Theorem 13, one can see that the δ_1 comes out to be much smaller than δ_f. This change in δ_1 results in a blowup of ε_1 and hence, the final privacy cost of the LT algorithm $(3\varepsilon_1 + 3\sqrt{2\delta_1})$, is much larger than what it would have been for learning one candidate without LT. We call this increase the *blowup* of privacy. We measure this blowup in Figure 5.1(left), for the setting of $\sigma = 4$, $L = 250$, $T = 10,000$ with varying dataset sizes (n). It can be seen that for $n = 5,000$, the blowup is 4.8x whereas for $n = 950,000$, the blowup is almost 7.3x (note the log scale on y-axis). Qualitatively similar trends persist for other choices of noise multiplier, lot size, and iterations. For example, in Figure 5.2, we run the LT and MA algorithms for $T = 10000$ with $\sigma = 4$ and $L = 250$ with varying candidate size and compare the final privacy costs. The γ value for the LT algorithm is set to $1/k$, where the k is the number of candidates. It can be seen that the privacy cost of LT (blue) remains almost constant for with increasing number of candidates. Figure 5.2 also demonstrates the exact number of candidates when the cost of MA (orange) remains below the LT cost. This insight is valuable in practice to a

practitioner to decide which algorithm to choose for hyperparameter tuning with respect to the number of candidates.



(a) 5k samples (b) 60k samples

Figure 5.2: Comparing LT vs MA with varying number of candidates at setting $\sigma = 4$, $L = 250$, $T = 10000$. MA can compose up to 14 and 26 candidates for the same cost of LT for dataset sizes 5k (left) and 60k (right), respectively.

Furthermore, we show that although LT entails a privacy blowup, decreasing γ (corresponding to training more individual learners with ε_1, δ_1) doesn't result in a significant difference in the final epsilon guaranteed by LT. In Figure 5.1(middle), we show the final epsilon cost for different dataset size and varying values of $\gamma \in [0.001, 0.01, 0.1, 1]$. It is interesting to note here that with smaller γ values, one can train many candidates (in expectation, $\frac{1}{\gamma}$) for negligible additional privacy cost. The blowup to train 1 candidate ($\gamma = 1$) versus $1,000$ candidates ($\gamma = 0.001$) increases from 33 to 39 for $n = 5,000$ and increases from 0.49 to 0.69, for $n = 950,000$. This increase is minimal in comparison to advanced composition, which grows proportional to $\mathcal{O}(\sqrt{k})$. However, another resource at play is the total training time, proportional to $1/\gamma$ (i.e., the total number of candidates). In summary, the LT algorithm is effective if an analyst has the privacy budget to afford the initial blowup, as the privacy cost of testing additional hyperparameters is insignificant.

Tuning cost via MA We learnt from the previous section that, LT permits selection from a large pool of hyperparameters (depending on the γ value) but incurs a constant privacy blowup. We compare LT with tuning using Moments Accountant (MA); for tuning via MA each hyperparameter candidate is trained by adding necessary Gaussian noise at each iteration, and the best hyperparameter candidate is selected at the end, MA is used as the composition mechanism for arriving at the final privacy cost of this process. We notice

that with the same initial privacy blowup of the LT algorithm, MA is able to compose a considerable number of hyperparameter candidates. In Figure 5.1(right), we show the number of candidates that can be composed using MA for the minimum privacy cost for running the LT algorithm ($\gamma = 1$), for the setting of $\sigma = 4$, $L = 250$, $T = 10,000$ and varying dataset size (n) on the x-axis. As the T and L is set constant, bigger n values in this graph correspond to fewer epochs of training and hence, worse utility. Depending on dataset size, MA can compose 14 candidates for $n = 5000$ and up to 175 candidates when $n = 100000$. It is perhaps surprising how well a standard composition technique performs versus LT. This information can be highly valuable to a practitioner who has limited privacy budget. Qualitatively similar trends persist for other choices of batch size, noise multiplier, and iterations.

From our experiments for both these tuning procedures, we conclude that while tuning with LT entails an initial privacy blowup, the additional privacy cost for trying more candidates (smaller γ) is minimal. Even though this has an additional computation cost, it can be appealing when an analyst wants to try numerous hyperparameters. On the other hand, for the same overall privacy cost, MA can be used to compose a significant number of hyperparameter candidates. Additionally, MA allows access to all intermediate learners, whereas LT allows access to only the final output parameters. In the sequel, this conclusion will be useful in making the naive MA approach a more appealing tool for some settings (e.g., tighter privacy budgets).

5.5 Tuning DP Optimizers

We detail aspects of tuning both non-adaptive and adaptive optimizers. We start with tuning non-adaptive optimizers. We theoretically and empirically demonstrate a connection between the learning rate and clipping threshold. We also establish that non-adaptive optimizers inevitably require searching over a large LR-clip grid to extract performant models. Adaptive optimizers forego this problem as they do not need to tune the hyperparameter dimension of learning rate. However, they introduce other hyperparameters that have known good choices in the non-private setting, and later we empirically show that they are good candidates in the private setting.

Tuning DP non-adaptive optimizers While many hyperparameters are restricted due to computational and privacy/utility targets, the learning rate α and the clipping threshold C have no a priori bounds. In what follows, we show an interplay between these parameters

by first theoretically analyzing the convergence of DPSGD. We then explore an illustrative experiment which demonstrates their entanglement. In the following theorem, we derive a bound on the expected excess risk of DPSGD and while doing so, show that the optimal learning rate, α_{opt}, is proportional to the inverse of C.

Theorem 14. *Let f be a convex and β-smooth function, and let $x^* = \underset{x\in\mathcal{S}}{\operatorname{argmin}} f(x)$. Let x_0 be an arbitrary point in $\mathcal{S}$, and $x_{t+1} = \Pi_{\mathcal{S}}(x_t - \alpha(g_t + z_t))$, where $g_t = \min(1, \frac{C}{\|\nabla f(x)\|^2})\nabla f(x)$ and $z_t \sim \mathcal{N}(0, \sigma^2 C^2)$ is the noise due to privacy. After T iterations, the optimal learning rate is $\alpha_{opt} = \frac{R}{CT\sqrt{1+\sigma^2}}$, where $\mathbb{E}[f(\frac{1}{T}\sum_i^T x_t) - f(x^*)] \leq \frac{RC\sqrt{1+\sigma^2}}{\sqrt{T}}$ and $R = \mathbb{E}[\|x_0 - x^*\|]$.*

Proof.

$$\begin{aligned}
\mathbb{E}[\|x_{t+1} - x^*\|^2 \;\big|\; x_t] &= \mathbb{E}[\|x_t - \alpha(g_t + z_t) - x^*\|^2 \;\big|\; x_t] \\
&= \mathbb{E}[\|x_t - x^*\|^2 - 2\alpha(g_t + z_t)(x_t - x^*) + \alpha^2\|(g_t + z_t)\|^2 \;\big|\; x_t] \\
&= \|x_t - x^*\|^2 - 2\alpha\mathbb{E}[(g_t + z_t) \;\big|\; x_t]^T(x_t - x^*) + \alpha^2\mathbb{E}[\|(g_t + z_t)\|^2 \;\big|\; x_t] \\
&\leq \|x_t - x^*\|^2 - 2\alpha[f(x_t) - f(x^*)] + \alpha^2\mathbb{E}[\|(g_t + z_t)\|^2 \;\big|\; x_t]
\end{aligned}$$

The inequality is due to convexity of the loss function and $\mathbb{E}[(g_t + z_t)] = g_t$ due to 0-mean noise. Taking expectation on both sides and reordering,

$$\begin{aligned}
2\alpha[f(x_t) - f(x^*)] &\leq \mathbb{E}[\|x_{t+1} - x^*\|^2] - \mathbb{E}[\|x_t - x^*\|^2] + \alpha^2\mathbb{E}[\|(g_t + z_t)\|^2 \\
&\leq \mathbb{E}[\|x_{t+1} - x^*\|^2] - \mathbb{E}[\|x_t - x^*\|^2] + \alpha^2(C^2 + C^2\sigma^2)
\end{aligned}$$

Summing for T steps and dividing both sides by $2\alpha T$,

$$\mathbb{E}[f(\frac{1}{T}\sum_i^T x_t) - f(x*)] \leq \frac{R^2}{2\alpha T} + \frac{\alpha C^2(1+\sigma^2)}{2} \tag{5.1}$$

Taking derivative and finding best value of α,

$$\alpha_{opt} = \frac{R}{C\sqrt{1+\sigma^2}T}$$

Plugging α_{opt} to Eq. 5.1,

$$\mathbb{E}[f(\frac{1}{T}\sum_i^T x_t) - f(x*)] \leq \frac{RC\sqrt{1+\sigma^2}}{\sqrt{T}}$$

□

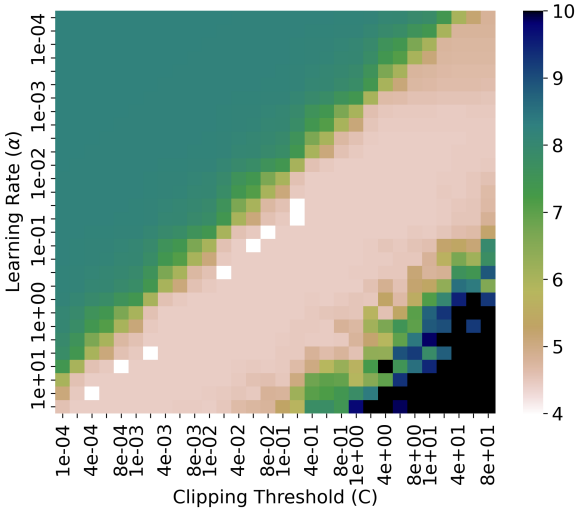


Figure 5.3: Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPSGD. The white pixels correspond to training losses below the 1st percentile. Note that most best loss values lie on a diagonal expressing the inverse connection between α and C.

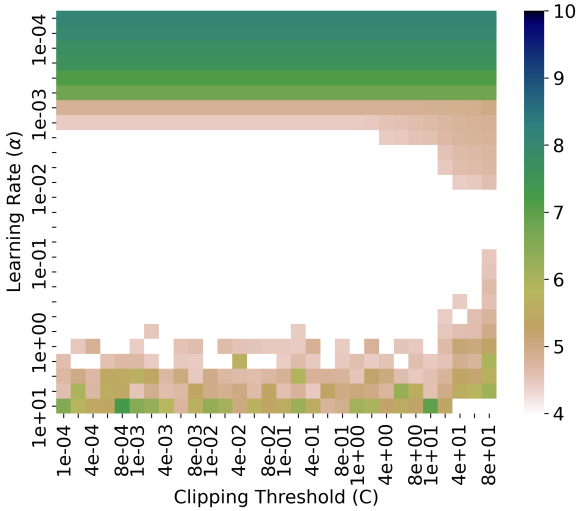


Figure 5.4: Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPAdam. The white pixels correspond to training loss lower than 1 percentile of DPSGD counterpart. Note that best loss values are dependent on α and spread over a wider range of C.

Theorem 14 gives a closed-form expression for the optimum learning rate for the convergence bound to hold in the convex setting. However, it is a function of the parameter R, which is unknown a priori to the analyst. Given constant T and σ, the optimal learning rate α_{opt} is inversely proportional to the clipping norm C. This is crucial information in practice because these parameters vary among datasets and are unbounded. This unboundedness property thus requires us to search over very large ranges of C and α when we have no prior knowledge of the dataset. It is natural to ask whether one can fix the clipping norm C and search only over a wide range for the learning rate α (or vice versa). We explore this relationship experimentally via simulations on a synthetic dataset as well as on the ENRON dataset, showing that fixing one of these two hyperparameters may often but not always result in an optimal model.

We train a linear regression model on a 10-dimensional synthetic dataset of input-label pairs (x, y) sampled from a distribution $\mathcal{D}$ as follows: $x_1, \ldots, x_d \sim \mathcal{U}(0, 1), y = x \cdot w^*, w^* = 10 \cdot \mathbf{1}^d$. We use the initialization $w_0 = \mathbf{0}^d$ and train for 5000 iterations. We set the initial weights of the model $w_0 = \mathbf{0}^d$, $d = 10$, $k = 10$ and train this linear regression model for 100 iterations. In the non-private setting, this model converges quickly with any reasonable learning rate, but in the private setting, we notice that the training loss

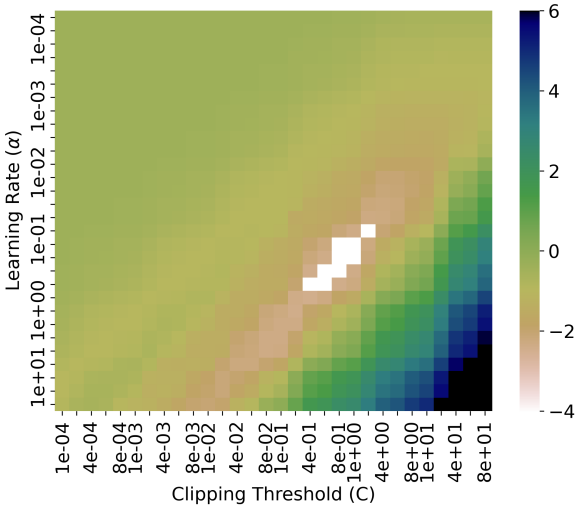


Figure 5.5: Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPSGD. The white pixels correspond to training losses below the 1st percentile. Note that most best loss values lie on a diagonal expressing the inverse connection between α and C.

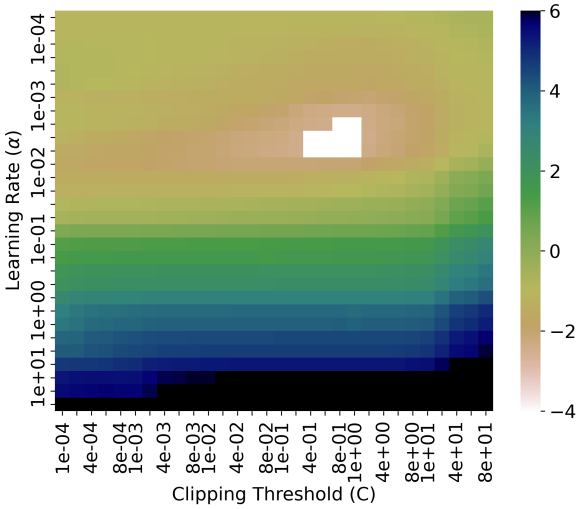
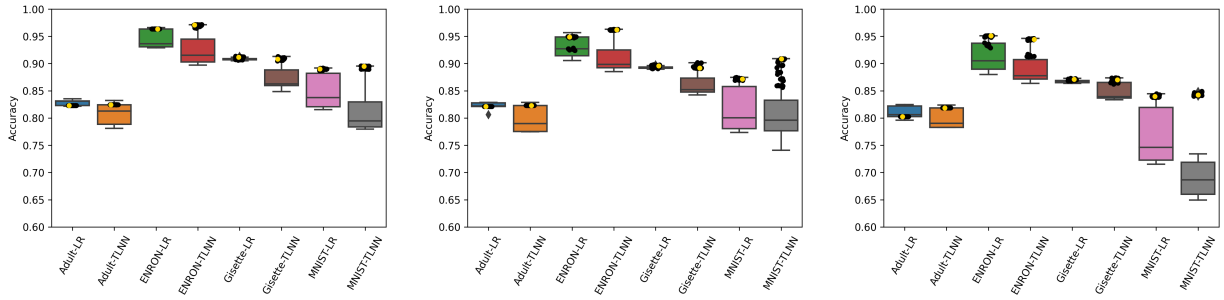


Figure 5.6: Log of training loss for simulation experiment at $\sigma = 4$ on a synthetic dataset with DPAdam. The white pixels correspond to training loss lower than 1 percentile of DPSGD counterpart. Note that best loss values are dependent on α and spread over a wider range of C.

depends heavily on the choice of α and C. Figure 5.3 shows a heat map for the log training loss when trained on (α,C) pairs taken from a large grid consisting of $[1, 2, 4, 5, 8]$ at scales of $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}]$. The best candidates (lowest 1 percentile of loss values) we demarcate with white pixels.

We observe two fundamental phenomena from this figure. First, to achieve the best accuracy, α and C need to be tuned on a large grid spanning several orders of magnitude for each of these parameters. Second, multiple (α,C) pairs achieve the best accuracy and all lie on the same diagonal, validating our theory for an inverse relation between learning rate and clipping norm. As mentioned earlier, one might hypothesize that by setting the clipping norm C constant and tuning α (corresponding to a vertical line in Figure 5.5) or vice versa, one could eliminate tuning a hyperparameter. However, note that not all C and α values correspond to the lowest loss. This phenomenon is evident by noticing that not all vertical or horizontal lines on this figure have white pixels. This happens, for example, at the extremes (e.g., at the top-right corner), but also for several intermediate and standard choices (e.g., $C = 0.1$ or 0.2). Again, the analyst has no way of knowing this a priori.

In Figure 5.4, we detail the results for the same simulation experiment with DPAdam



(a) $\sigma = 2$ (b) $\sigma = 4$ (c) $\sigma = 8$

Figure 5.7: Ranking hyperparameter candidates across datasets. The black points correspond to the candidates with $\alpha = 0.001$ (with all permutations of β_1, β_2 from our search grid); the gold corresponds to the candidate with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$.

(with default Adam hyperparameters) as the underlying optimizer. Here we notice that the inverse relation between α and C no longer hold as we intuited earlier. In order to be able to compare these two figures, we mark all candidates with lower losses than that of the lowest 1 percentile of Figure 5.3 with white in Figure 5.4. Figure 5.4 suggests that there is a small range of ideal choices for the initial learning rate α, and within these good choices of the α, DPAdam is significantly robust to the choice of clip value, as we see white pixels for a large range of clip values that encompass clip values used in practice. We repeat the same experiment over the ENRON dataset instead of our synthetic dataset, and observe similar trends as seen in Figure 5.5 and Figure 5.6. We conclude that to privately tune non-adaptive optimizers, we require a larger grid of hyperparameter options than their adaptive counterparts.

Tuning DP adaptive optimizers Adaptive optimizers automatically adapt over the learning rate α, requiring us to tune only over the clipping norm C. But recall our key question: can we train models that perform competitively with the fine-tuned counterparts from DPSGD?

Adam [94], the canonical adaptive optimizer introduces two new hyperparameters, which are the first and second moment exponential decay parameters (β_1 and β_2). In the non-private setting, these parameters are relatively insensitive, and default values of $\alpha = 0.001, \beta_1 = 0.9$, and $\beta_2 = 0.999$ are recommended based on empirical findings, requiring no additional tuning for this hyperparameter triple. Hence before we compare DPAdam and DPSGD, we first find and establish such recommended values for this hyperparameter triple in the DP setting next, and then show that DPAdam with a small hyperparameter space performs competitively with DPSGD in Section 5.6.

To establish default choices of α, β_1, and β_2 for DPAdam, we evaluate this private

Table 5.1: Datasets used in experiment

Dataset	Type	#Samples	#Dims	#Classes
MNIST	Image	70000	784	10
Gisette	Image	6000	5000	2
Adult	Structured	45222	202	2
ENRON	Structured	5172	5512	2

optimizer over four diverse datasets Table 5.1, and two learning models including logistic regression and a neural network with one 100 neurons hidden layer (TLNN). These selected datasets include both low-dimensional data (where the number of samples greatly outnumbers the dimensionality) and high-dimensional data (where the number of samples and dimensionality are at same scale). Since we still have a large hyperparameter space to tune over, for the rest of this work, we fix a constant lot size ($L = 250$), and consider tuning over three different noise levels, $\sigma \in [2, 4, 8]$, so that we can study the effects of tuning the other hyperparameters more thoroughly. All experiments are repeated three times and averaged before reporting. Additionally, in this particular experiment since we focus on α, β_1, and β_2, we also fix the clipping threshold $C = 0.5$, and $T = 2500$ iterations of training. For each dataset and model, we run DPAdam three times with hyperparameters $(\alpha, \beta_1, \beta_2)$ from the grids, $\alpha \in [0.001, 0.05, 0.01, 0.2, 0.5]$, $\beta_1, \beta_2 \in [0.8, 0.85, 0.9, 0.95, 0.99.0.999]$.

We show that the default hyperparameter choice $(\alpha, \beta_1, \beta_2)$ of Adam in the non-private setting also works well for DPAdam. Figure 5.7 shows the boxplots of testing accuracies of DPAdam over the different hyperparameter choices. When α is 0.001 (same as in the non-private setting), all the datasets and models have final testing accuracies (marked in black) close to the best possible (and in most cases it is in fact the best) accuracy. Furthermore, we also highlight the accuracy of the suggested default choice ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) using gold dots. Hence, for the ease of using DPAdam, we suggest the non-private default values for these parameters in the private setting as well and hence in all our subsequent experiments.

5.6 Advantages of Tuning using DPAdam

In the non-private setting, adaptive optimizers like Adam enjoy a smaller hyperparameter tuning space than SGD. We ask two questions in this section. First, can DPAdam (with little tuning) achieve accuracy comparable to a well-tuned DPSGD? Second, what is the privacy-accuracy tradeoff one incurs when using either of the two methods we detail in Section 5.4 for hyperparameter selection.

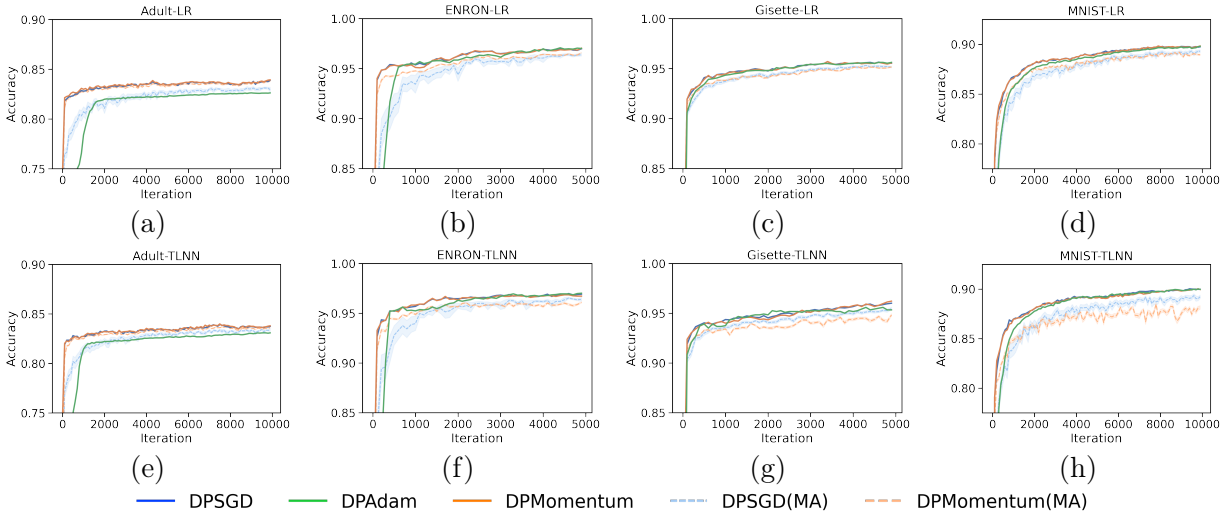


Figure 5.8: Comparing the testing accuracy curves of DPAdam, DPSGD and DPMomentum models across their hyperparameter tuning grids with $\sigma = 4$. The limits for y-axis are adjusted based on the dataset while maintaining a 15% range for all.

To answer both questions, we compare DPAdam and DPSGD over the same set of datasets and models from the previous section. We report the accuracy of DPSGD with a range of learning rates and clipping values shown in Table 5.2, and the testing accuracy of DPAdam with default parameter choice from Section 5.5 ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) and a range of clipping values C in Table 5.2. In total, DPSGD has 40 candidates to tune over, and DPAdam has 4. This is because we have shown in Section 5.5 that DPSGD needs a wide grid to obtain the best accuracy when data distributions are unknown. Additionally, we also consider the DPMomentum optimizer. Similar to how we searched for default tuning choices for DPAdam in Section 5.5, we investigate if there exists a qualitatively good choice for the momentum hyperparameter, and unfortunately our results show that there is no such choice.

In order to show the comparison from both sides of the privacy-accuracy tradeoff, we compare the three optimizers through i) the privacy cost when extracting the best accuracy from these optimizers, and ii) the accuracy one would obtain from them under the tight privacy constraints.

Prioritizing Accuracy We show experiments for $\sigma = 4$ in Figure 5.8, results for other values of σ are displayed in the full paper [126]. For each dataset and model, we train three times for each hyperparameter candidate and report the max every 100 iterations, corre-

Table 5.2: Parameter grid for comparing DPSGD and DPAdam

Optimizer	Parameter	Values
DPSGD	α	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1
	C	0.1, 0.2, 0.5, 1
DPMomentum	α	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1
	C	0.1, 0.2, 0.5, 1
	m	0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99
DPAdam	C	0.1, 0.2, 0.5, 1

sponding to the dark lines for each optimizer. We note that their maxima are extremely similar. However, Table 5.3 shows the final privacy costs incurred by each of these max accuracy lines, and reflects our claims from Section 5.4 that using fewer hyperparameter candidates and composing privacy via MA gives a much tighter privacy guarantee.

Dataset	DPSGD	DPMomentum	DPAdam
Adult	5.01	5.23	1.91
ENRON	30.86	32.31	12.80
Gisette	26.40	27.64	10.76
MNIST	3.01	3.14	1.14

Table 5.3: Final ε (at $\delta = 10^{-6}$) for optimizers for the LR Models (Figure 5.8). DPSGD and DPMomentum use LT for privacy accounting; DPAdam uses MA.

Prioritizing Privacy Additionally in Figure 5.8, DPSGD and DPMomentum have pastel dotted lines corresponding to their mean accuracy attained using the MA composition that provides the tightest privacy guarantees for DPAdam. These pastel lines are the mean accuracies (with 95% CI) from 100 repetitions of this experiment. Since DPAdam has only 4 hyperparameter candidates, for this experiment, we sample 4 of the candidates at random for DPSGD and DPMomentum so that they all incur the same privacy cost. Since the candidate pool is significantly larger for DPSGD and DPMomentum, we additionally scrutinize the parameter grid for them and prune the learning rates that perform poorly. Our pruning process (detailed in the supplement) is quite generous, and favours minimizing the hyperparameter space of DPSGD and DPMomentum as much possible.[1] Despite

[1]Note, pruning itself is of course unfair; the intent was to design a DP optimizer that can be used on any data distributions that we have no prior knowledge of. To do so with DPSGD one would have to consider a significantly wide range of (α, C) pairs to cover 'good' candidates as we illustrated in Section 5.5

the pruning advantage we see that these optimizers perform subpar than DPAdam when constrained with privacy.

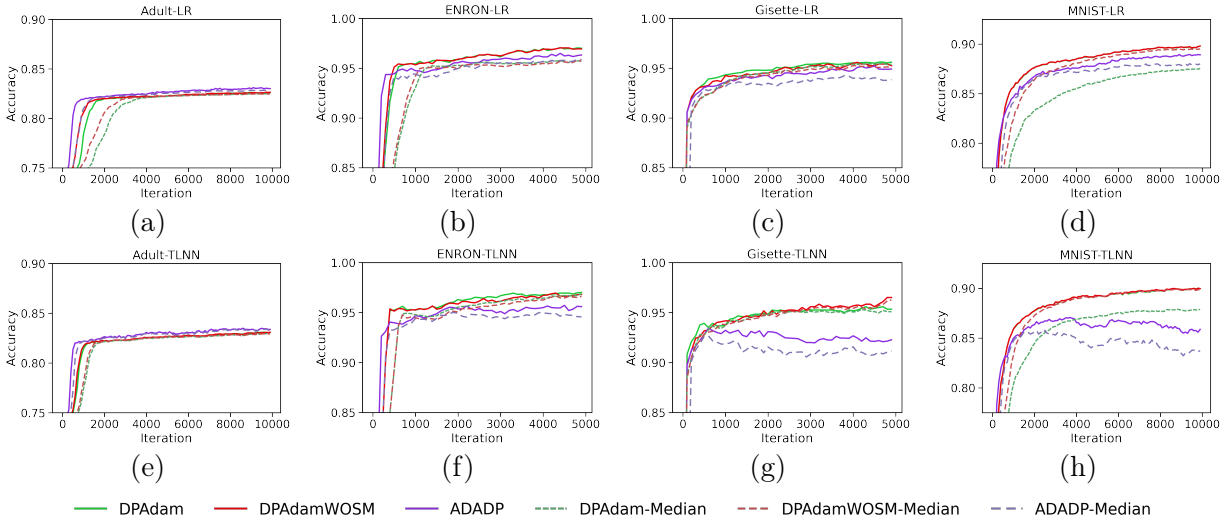


Figure 5.9: Comparing the testing accuracy curves of DPAdam, ADADP and DPAdamWOSM models across the hyperparameter tuning grid from Table 5.2 with $\sigma = 4$. The y-axes limits are adjusted based on the dataset while maintaining a 15% range for all.

5.7 DPAdamWOSM

In addition to a decaying average of past gradient updates, DPAdam also maintains a decaying average of their second moments. In this section, we design DPAdamWOSM (DPAdam Without Second Moments), a new DP optimizer that operates only using a decaying average of past gradients, as well as eliminates the need to tune the learning rate parameter. We achieve this by analyzing the convergence behavior of the second-moment decaying average in DPAdam in regimes where the scale of noise added is much higher than the scale of the clipped gradients. Setting the *effective step size* (ESS) of DPAdam to the converged constant, and removing all computations related to the second-moment updates, results in DPAdamWOSM. We empirically demonstrate that DPAdamWOSM matches the utility of DPAdam, while requiring less computation than DPAdam.

Observe that removing the second-moment updates from DPAdam reduces it to DPMomentum with one additional feature: bias-correction to the first-moment decaying average,

Algorithm 11: Optimization using DPAdamWOSM

Require: Training set $A : \{x_1, ..., x_n\}$, Loss function $\mathcal{L}(\theta)$, Parameters: Lot size L, Learning rate α, Gradient norm bound C, Noise scale σ, Total number of iterations T, Exponential decay rate β_1

Initialize model with θ_0 randomly;
Initialize first moment vector $m_0 = 0$;
Set learning rate to ESS $\alpha = \frac{10^{-3}}{(\sigma C/L)+10^{-8}}$;
for $t \in [1, T]$ **do**
    Sample a random subset $L_t \subseteq A$, by independently including each element of A with probability L/n;
    Compute gradient $\forall x_i \in L_t$
    $g_t(x_i) = \nabla_\theta \mathcal{L}(\theta_t, x_i)$;
    Clip each gradient in ℓ_2 norm to C $\bar{g}_t(x_i) = g_t(x_i)/\max(1, \frac{\|g_t(x_i)\|_2}{C})$;
    Add noise $\tilde{g}_t = \frac{1}{|L|}(\sum_i \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I))$;
    Exponentially average the first moment
    $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \tilde{g}_t$
    Perform bias correction
    $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$;
    Update model $\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t$;
end for
Compute privacy cost using Moments Accountant.

which DPAdam does to account for its initialization at the origin. While the resultant optimizer still requires tuning the learning rate (in addition to other hyperparameters like the clipping threshold), DPAdamWOSM can be viewed as self-tuning the learning rate by fixing it to the converged effective step size in DPAdam.

Effective step size (ESS) in DPAdam As we saw in Figure 5.8, DPAdam results have less variance than DPSGD due to its adaptive learning rate. To understand this phenomenon better, we inspect DPAdam's update step. DPAdam being an adaptive optimizer picks per-parameter ESS as $\frac{\alpha}{\sqrt{\hat{v}_t}+\xi}$, which is the base learning rate α scaled by the second moment of the individual parameter gradients. The variable ξ is adapted from the Adam optimizer and ensures that the denominator does not get to 0. We notice that when $g \to 0$, the ESS for DPAdam converges for the first moment gradient, which innately accounts for the clip bound one is training with. This may happen at later iterations, when the model is close to its minima and the gradients get close to zero.

Theorem 15. *The effective step size (ESS) for DPAdam with $g \to 0$ converges to $ESS^* = \frac{\alpha}{(\sigma C/L)+\xi}$.*

Proof. Recall that the average noisy gradient over a lot is $\tilde{g} = g + \mathcal{N}(0, \sigma^2 C^2)/L$. We now look at the effect of this noisy gradient on the effective step size (ESS) of DPAdam. As $g \to 0$, the second moment of DPAdam converges to $\frac{\sigma^2 C^2}{L^2}$. This gives us the converging value for ESS:

$$ESS^* = \frac{\alpha}{\sqrt{\hat{v}_t} + \xi} = \frac{\alpha}{\sqrt{\frac{\sigma^2 C^2}{L^2}} + \xi} = \frac{\alpha}{(\sigma C/L) + \xi}$$

□

Theorem 15 gives a closed form expression that ESS converges to. We can use this value in place of $\frac{\alpha}{\sqrt{\hat{v}_t}+\xi}$ in the update step from the inception of the learning process. Since the second-moment updates (e.g., $\hat{v}_t$) are not used anymore, removing them results in our new optimizer DPAdamWOSM. The pseudocode for DPAdamWOSM is shown in Algorithm 11.

Comparing Adaptive Optimizers We evaluate DPAdamWOSM by running it alongside DPAdam and ADADP with the same hyperparameter grid in the appendix. For brevity, we show experiments on $\sigma = 4$ and others appear in the supplement. In Figure 5.9, we show the maximum and median accuracy curves for all the optimizers. We display the median accuracy curves (shown in dotted), as an indicator of the quality of the entire pool of hyperparameter candidates for a given optimizer; which in this case is strictly over the choices of clip. The max lines for ADADP lies beneath DPAdam and DPAdamWOSM for all dataset except Adult. Also, the max accuracy line for DPAdamWOSM runs alongside DPAdam which means that it can perform as good as DPAdam throughout training. The median line for DPAdamWOSM also performs alongside DPAdam and in some cases is able to beat it (e.g, the median for DPAdamWOSM for MNIST-LR and MNIST-TLNN lies above the median line of DPAdam). This occurrence is seen because DPAdamWOSM uses the converged ESS from the first iteration of training.

Chapter 6

Related Work

In this chapter, we provide a systematic review of related work for this thesis. Although there have been attempts at solving some of the challenges put out in this thesis, this thesis provides the first attempt at combining these challenges into a single end-to-end framework. In what follows, we provide related works for each contribution separately.

6.1 Differential Privacy and Data Repair

Various classes of constraints have been proposed by the literature, including FDs, conditional FDs [27], metric FDs [98], and DCs [37]. Our work focuses on DCs, a general class of integrity constraints that subsumes all of the aforementioned constraints. While computing the minimal data repair in some cases has been shown to be polynomial time [113], computing the minimal repair in most general cases, corresponding to $\mathcal{I}_{\mathsf{R}}$, is NP-hard. Therefore, a prominent vein of research has been devoted to utilizing these constraints for data repairing [10, 153, 39, 23, 57, 113, 68, 65]. The repair model in these works varies between several options: tuple deletion, cell value updates, tuple addition, and combinations thereof. The process of data repairing through such algorithms can be time consuming due to the size of the data and the size and complexity of the constraint set. Hence, previous work [110, 114] proposed to keep track of the repairing process and ensure that it progresses correctly using inconsistency measures. In our work, we capitalize on the suitability of these measures to DP as they provide an aggregate form that summarizes the quality of the data for a given set of constraints.

DP has been studied in multiple settings [86, 97, 182, 85, 116, 167], including systems that support complex SQL queries that, in particular, can express integrity constraints [47]. The utilization of graph databases under DP has also been thoroughly explored, with both edge privacy [75, 89, 90, 157, 91, 194] and node privacy [92, 25, 42]. Our approach draws on [42] to allow efficient DP computation of the inconsistency measures over the conflict graph. In contrast, we have seen worse performance from alternative approaches for releasing graph statistics that tend to truncate edges or nodes aggressively. In the context of data quality, previous work [99] has proposed a framework for releasing a private version of a database relation for publishing, supporting specific repair operations, while more recent, work [64] provides a DP synthetic data generation mechanism that considers soft DCs [37].

6.2 Differentially Private Synthetic Data Generation for Missing Values

Differentially private synthetic data generation has been studied vastly in prior literature [28, 58, 199, 135]. Prior works generate synthetic data via statistical approaches which estimate low-dimensional marginal distributions [150, 185], deep learning approaches [61, 87], or the combination of the two [62]. However, all of these algorithms focus on the no missing data setting. Some prior work look into missing data imputation for private datasets but are either not in the differential privacy setting [143, 79, 82] or do not support generating synthetic data as a part of their work [40, 100, 41]. Patki et al. [143] propose the synthetic data vault framework, which identifies and repairs inconsistencies in the generated synthetic data from incomplete data but does not consider any privacy guarantees. Their approach uses low-way marginals and learns them using Gaussian copulas that may be enhanced using our partial marginal observation approach if made private. Huang et al. [79] and Jagannathan et al. [82] imputation of missing data as a cleaning algorithm for private datasets but consider privacy definitions of k-anonymity and cryptographic distributive computing, respectively. Other works focus on privately imputing missing values. The existing differentially private solutions [100, 40, 41] focus on data imputation and are difficult to adapt for the synthetic data generation problem. PrivateClean [100] requires a human in the loop that can specify the user-defined specific imputation functions to clean the database. The k-means based word by Clifton et al. [40] and imputation first approach in Das et al. [41] use part of the privacy budget to impute the missing values using OLS regression. We note that such OLS regression and k-means models cannot be applied to categorical attributes, and using them for imputation incurs a significant amount of privacy budget and leaves little budget for data synthesis.

6.3 Hyperparameter Tuning with Differential Privacy

Hyperparameter tuning plays a vital role in machine learning practice. In the non-private setting, ML practitioners use grid search, random search, Bayesian optimization techniques [165] or AutoML [76] techniques to tune their models. However, there hasn't been much research on private hyperparameter tuning procedures due to the significant associated privacy costs. Each set of hyperparameter configuration results in a privacy-utility tradeoff. This tradeoff for multiple configurations can be captured by Pareto frontiers using multivariate Bayesian optimization over parameter dimensions [16]. However, this method asks the model curator to query the dataset multiple times which requires non-private access to the dataset. There have been some end-to-end private tuning procedures [31, 32, 101] which work for a selected number of hyperparameter sets. These results work either in restricted settings for few combinations of candidates under relaxations of approximate differential privacy. The most relevant work to ours is an approach for private selection from private candidates [109]. Their work provides two methods, one which outputs a candidate with accuracy greater than a given threshold, and another which randomly stops and outputs the best candidate seen so far. The first approach is of limited utility in practice as it requires a prior accuracy bound for the dataset. The second variant incurs a considerable overhead in privacy cost. We study the second approach and compare it with a naive approach based on Moments Accountant [5] which would scale as the square root of the number of candidates. Recent work shows generalized version of the approach for private selection on diverse with better bounds using Rényi DP [140].

Chapter 7

Conclusion

In this thesis, we identified gaps in the practice of differentially private learning. We specified that multiple challenges could exist when the model curator is not trusted and the private learning framework has a limited privacy budget. These challenges stem from performing various tasks such as data exploration, data cleaning, and hyperparameter tuning within the restrictions. To systematically solve these challenges, we presented an end-to-end differentially private framework that begins from data collection and has a differentially private output, such as a private model or generated synthetic data. We took three significant steps to build this framework.

7.1 Contributions

Our first contribution shown in Chapter 3 was to measure data quality of a private dataset. We proposed a new problem of studying inconsistency measures for a private dataset in the DP setting. We studied five measures and showed that two are intractable with DP, and the other three face a significant challenge of high sensitivity. To solve this challenge, we leveraged the dataset's corresponding conflict graph and used graph projection and a novel approximate DP vertex cover size algorithm to accurately estimate the private inconsistency measures. We found that parameter selection was a significant challenge and were able to overcome it using optimization techniques based on the constraint set. To test our algorithm, we experimented with five real-world datasets with varying density properties and showed that our algorithm could accurately calculate these measures across all datasets. This work significantly contributes to the end-to-end system as a first step in data exploration. Once the model curator has an idea about the data quality of the private

dataset, they can plan their next steps in the private learning framework. For example, if the dataset has specific errors, they would spend more time and privacy budget in repairing those errors; otherwise, they can directly move on to the model training step.

The second contribution as in Chapter 4 presented a comprehensive study on differentially private synthetic data generation algorithms for private datasets with missing values. Our proposed adaptive recourse methods outperformed classical approaches and struck a balance between privacy and utility. We also provided techniques for calculating privacy bounds and demonstrated the effectiveness of our methods through extensive experiments on real-world datasets. The findings of this contribution have essential implications for the proposed end-to-end framework in this thesis. Using techniques in this work, if the model curator knows missing values in the dataset, they can quickly adapt the existing DP model training techniques to develop better quality synthetic data.

In the final contribution in Chapter 5, we thoroughly investigated honest hyperparameter selection for DP optimizers. We compared two existing private methods (LT and MA) to search for hyperparameter candidates. We showed that the former incurs a significant privacy cost but can compose over many candidates, while the latter is effective when the number of candidates is small. Next, we explored connections between the clipping norm and the step size hyperparameter to show their inverse relationship. Additionally, we compared non-adaptive and adaptive optimizers, demonstrating that the latter typically achieves more consistent performance over various hyperparameter settings. This can be vital for applications with scarce public data, resulting in difficulties when tuning hyperparameters. Finally, we revealed that DPAdam converges to a static learning rate when the noise dominates the gradients. This insight allowed us to derive a novel optimizer DPAdamWOSM, a variant of DPAdam which avoids the second-moment computation and enjoys better accuracy, especially at earlier iterations. The insights from our paper contribute to the end-to-end framework by providing the model curator crucial insights into the choice of hyperparameters and optimizer that they should use for differentially private training of machine learning models.

7.2 Future Work

The work presented in the thesis has many limitations, leading to multiple interesting future directions. We conclude this thesis by enumerating some future work.

Identifying errors and private data repair. In Chapter 3, we show how we can compute the total number of errors/violations as a data quality metric using inconsistency

measures. This data quality metric, however, does not talk about the identity of the rows participating in these violations. For example, the repair measure outputs the total number of rows that should be repaired so that there are no more violations in the dataset, but does not return which rows. This is because our algorithm for the repair measure only outputs the size of the vertex cover, not the vertex cover itself. It would be a nice idea to work further on this so that the vertex cover can be computed. This can be done by developing an approximation algorithm for vertex cover [26].

Preserving missing values in synthetic data generation. In Chapter 4, we showed adaptive methods to work with missing data in differentially private synthetic data generation. This work aims to generate synthetic data without any missing values as output. However, an interesting direction could also be to generate synthetic data with missing values. The missing values can then be treated as a post-processing step. One way to do this could be to formulate missing data as integrity constraints and preserve these integrity constraints as part of the synthetic data generation with Kamino [62]. To do so, however, we must discover these constraints between missing and observed values in the dataset. Fortunately, some interesting work in this line has discovered such dependencies without privacy [81, 197]. In future work, we plan to look further into these works and develop differentially private algorithms to discover these constraints and aim to preserve them in the synthetic data generation process.

Extending tuning to larger models. We are now witnessing a swift growth in the size of machine learning models and tasks. The latest models have grown up to billions of parameters and are now being used for complicated tasks such as video generation and language reasoning. Training these large models is an arduous but promising direction. In our work in Chapter 5, we show how the optimizer of a private neural network can be tuned, but more work needs to be done to extend differential privacy to large models. Most large models are based on complicated neural network architectures such as transformers [178] and mixture-of-expert [174]. In the future, we plan to look closely into these layers and training methods such as parameter-efficient fine-tuning [45] to achieve differentially private reality for large models.

References

[1] Data never sleeps 10.0. https://www.domo.com/data-never-sleeps. Accessed: 2025-05-13.

[2] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *OJ*, 2016-04-27.

[3] Holoclean code, Online.

[4] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM.

[5] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, 2016.

[6] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318. ACM, 2016.

[7] Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Latanya Sweeney. Privacy preserving synthetic data release using deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 510–526. Springer, 2018.

[8] John M Abowd. The us census bureau adopts differential privacy. In *SIGKDD*, pages 2867–2867, 2018.

[9] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1109–1121, 2018.

[10] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: Algorithms and complexity. In *ICDT*, pages 31–41, 2009.

[11] Anish Agarwal and Rahul Singh. Causal inference with corrupted data: Measurement error, missing values, discretization, and differential privacy. *arXiv preprint arXiv:2107.02780*, 2021.

[12] Wael Alghamdi, Shahab Asoodeh, Flavio P Calmon, Juan Felipe Gomez, Oliver Kosut, Lalitha Sankar, and Fei Wei. The saddle-point accountant for differential privacy. *arXiv preprint arXiv:2208.09595*, 2022.

[13] Rebecca R Andridge and Roderick JA Little. A review of hot deck imputation for survey non-response. *International statistical review*, 78(1):40–64, 2010.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[15] Brooke Auxier, Lee Rainie, Monica Anderson, Andrew Perrin, Madhu Kumar, and Erica Turner. Americans and privacy - concerned confused and feeling lack of control over their personal information. *Pew Research Center*, 2019.

[16] Brendan Avent, Javier González, Tom Diethe, Andrei Paleyes, and Borja Balle. Automatic discovery of privacy–utility pareto fronts. *Proceedings on Privacy Enhancing Technologies*, 2020(4):5–23, 2020.

[17] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in Neural Information Processing Systems*, 31, 2018.

[18] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems 31*, NeurIPS '18, pages 6277–6287. Curran Associates, Inc., 2018.

[19] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, pages 273–282, 2007.

[20] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473, 2014.

[21] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[22] Leopoldo E. Bertossi. Measuring and computing database inconsistency via repairs. In *SUM*, volume 11142 of *LNCS*, pages 368–372. Springer, 2018.

[23] Leopoldo E. Bertossi, Solmaz Kolahi, and Laks V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *Theory Comput. Syst.*, 52(3):441–482, 2013.

[24] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. Efficient denial constraint discovery with hydra. *Proc. VLDB Endow.*, 11(3):311–323, 2017.

[25] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[26] Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity. *arXiv preprint arXiv:2210.03831*, 2022.

[27] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, 2007.

[28] Claire McKay Bowen and Fang Liu. Comparative study of differentially private data synthesis methods. *Statistical Science*, 35(2):280–307, May 2020.

[29] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 634–649. IEEE, 2015.

[30] Thee Chanyaswad, Changchang Liu, and Prateek Mittal. Ron-gauss: Enhancing utility in non-interactive private data release. *Proceedings on Privacy Enhancing Technologies*, 2019(1):26–46, 2019.

[31] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[32] Kamalika Chaudhuri and Staal A Vinterbo. A stability-based validation procedure for differentially private machine learning. *Advances in Neural Information Processing Systems*, 26:2652–2660, 2013.

[33] R. Chawla. Deepfakes : How a pervert shook the world. *International Journal for Advance Research and Development*, 4:4–8, 2019.

[34] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. *Advances in Neural Information Processing Systems*, 33:12673–12684, 2020.

[35] Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaafar, and Haojin Zhu. Differentially private data generative models. *CoRR*, abs/1812.02274, 2018.

[36] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *SIGKDD*, pages 129–138, 2015.

[37] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

[38] Xu Chu, Ihab F Ilyas, and Paolo Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013.

[39] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *ICDE*, pages 458–469, 2013.

[40] Chris Clifton, Eric J. Hanson, Keith Merrill, and Shawn Merrill. Differentially private k-nearest neighbor missing data imputation. *ACM Trans. Priv. Secur.*, 25(3):16:1–16:23, 2022.

[41] Soumojit Das, Jorg Dreschler, Keith Merrill, and Shawn Merrill. Imputation under differential privacy. *CoRR*, abs/2206.15063, 2022.

[42] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *SIGMOD*, pages 123–138, 2016.

[43] Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale, 2022.

[44] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *Advances in Neural Information Processing Systems*, 30, 2017.

[45] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[46] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. Differentially private diffusion models. *Transactions on Machine Learning Research*, 2023.

[47] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *SIGMOD*, pages 759–772, 2022.

[48] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[49] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[50] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP*, volume 4052, pages 1–12, 2006.

[51] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004, pages 486–503. Springer, 2006.

[52] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[53] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, Berlin, Heidelberg, 2006. Springer.

[54] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.

[55] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[56] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[57] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Dichotomies in the complexity of preferred repairs. In *PODS*, pages 3–15, 2015.

[58] Liyue Fan. A survey of differentially private generative adversarial networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[59] Mei Ling Fang, Devendra Singh Dhami, and Kristian Kersting. Dp-ctgan: Differentially private medical data generation using ctgans. In *International Conference on Artificial Intelligence in Medicine*, pages 178–188. Springer, 2022.

[60] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[61] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *SEC*, pages 151–164, 2019.

[62] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F Ilyas. Kamino: Constraint-aware differentially private data synthesis. *Proceedings of the VLDB Endowment*, 14(10):1886–1899, 2021.

[63] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. Kamino: Constraint-aware differentially private data synthesis. *Proc. VLDB Endow.*, 14(10):1886–1899, 2021.

[64] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. Kamino: Constraint-aware differentially private data synthesis. *Proc. VLDB Endow.*, 14(10):1886–1899, 2021.

[65] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. The LLUNATIC data-cleaning framework. *Proc. VLDB Endow.*, 6(9):625–636, 2013.

[66] Kylie Getz, Rebecca A Hubbard, and Kristin A Linn. Performance of multiple imputation using modern machine learning methods in electronic health records data. *Epidemiology*, pages 10–1097, 2023.

[67] Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L Smith, Olivia Wiles, and Borja Balle. Differentially private diffusion models generate useful synthetic images. *arXiv preprint arXiv:2302.13861*, 2023.

[68] Amir Gilad, Daniel Deutch, and Sudeepa Roy. On multiple semantics for declarative database repairs. In *SIGMOD*, pages 817–831, 2020.

[69] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021.

[70] John Grant and Anthony Hunter. Distance-based measures of inconsistency. In *ECSQARU*, volume 7958 of *LNCS*, pages 230–241. Springer, 2013.

[71] John Grant and Anthony Hunter. Semantic inconsistency measures using 3-valued logics. *Int. J. Approx. Reason.*, 156:38–60, 2023.

[72] Jerrold R. Griggs, Charles M. Grinstead, and David R. Guichard. The number of maximal independent sets in a connected graph. *Discret. Math.*, 68(2-3):211–220, 1988.

[73] Rahul Gupta. Data augmentation for low resource sentiment analysis using generative adversarial networks. In *ICASSP*, pages 7380–7384. IEEE, 2019.

[74] R. Gylberth, R. Adnan, S. Yazid, and T. Basaruddin. Differentially private optimization algorithms for deep neural networks. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 387–394, 2017.

[75] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.

[76] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[77] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[78] Justin Hsu, Aaron Roth, Tim Roughgarden, and Jonathan Ullman. Privately solving linear programs. In *ICALP*, pages 612–624. Springer, 2014.

[79] Yu Huang, Mostafa Milani, and Fei Chiang. Pacas: Privacy-aware, data cleaning-as-a-service. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1023–1030. IEEE, 2018.

[80] IBM. Cost of a data breach report. 2020.

[81] Ihab F Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. Cords: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 647–658, 2004.

[82] Geetha Jagannathan and Rebecca N Wright. Privacy-preserving imputation of missing data. *Data & Knowledge Engineering*, 65(1):40–56, 2008.

[83] Seung Jae Jeon, Myung Seok Go, and Ju Hyun Namgung. Use of personal information for artificial intelligence learning data under the personal information protection act: the case of lee-luda, an artificial-intelligence chatbot in south korea. *Asia Pacific Law Review*, 31(1):55–72, 2023.

[84] José M Jerez, Ignacio Molina, Pedro J García-Laencina, Emilio Alba, Nuria Ribelles, Miguel Martín, and Leonardo Franco. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial intelligence in medicine*, 50(2):105–115, 2010.

[85] Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[86] Noah M Johnson, Joseph P Near, and Dawn Xiaodong Song. Practical differential privacy for sql queries using elastic sensitivity. *CoRR, abs/1706.09479*, 2017.

[87] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: generating synthetic data with differential privacy guarantees. In *ICLR*, 2019.

[88] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[89] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[90] Vishesh Karwa and Aleksandra B Slavković. Differentially private graphical degree sequences and synthetic graphs. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2012, Palermo, Italy, September 26-28, 2012. Proceedings*, pages 273–285. Springer, 2012.

[91] Vishesh Karwa, Aleksandra B Slavković, and Pavel Krivitsky. Differentially private exponential random graphs. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2014, Ibiza, Spain, September 17-19, 2014. Proceedings*, pages 143–155. Springer, 2014.

[92] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Analyzing graphs with node differential privacy. In Amit Sahai, editor, *TCC*, volume 7785, pages 457–476. Springer, 2013.

[93] Benny Kimelfeld, Ester Livshits, and Liat Peterfreund. Counting and enumerating preferred database repairs. *Theor. Comput. Sci.*, 837:115–157, 2020.

[94] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[95] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[96] Antti Koskela and Antti Honkela. Learning rate adaptation for differentially private learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2465–2475. PMLR, 2020.

[97] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment*, 12(11):1371–1384, 2019.

[98] Nick Koudas, Avishek Saha, Divesh Srivastava, and Suresh Venkatasubramanian. Metric functional dependencies. In *ICDE*, 2009.

[99] Sanjay Krishnan, Jiannan Wang, Michael J. Franklin, Ken Goldberg, and Tim Kraska. PrivateClean: Data cleaning and differential privacy. In *SIGMOD*, pages 937–951. ACM, 2016.

[100] Sanjay Krishnan, Jiannan Wang, Michael J Franklin, Ken Goldberg, and Tim Kraska. Privateclean: Data cleaning and differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 937–951, 2016.

[101] Matt Kusner, Jacob Gardner, Roman Garnett, and Kilian Weinberger. Differentially private bayesian optimization. In *International conference on machine learning*, pages 918–927. PMLR, 2015.

[102] Kamakshi Lakshminarayan, Steven A Harp, Robert P Goldman, Tariq Samad, et al. Imputation of missing data using machine learning techniques. In *KDD*, volume 96, 1996.

[103] Haoran Li, Li Xiong, Lifan Zhang, and Xiaoqian Jiang. Dpsynthesizer: Differentially private data synthesizer for privacy preserving data sharing. *Proc. VLDB Endow.*, 7(13):1677–1680, 2014.

[104] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. Misgan: Learning from incomplete data with generative adversarial networks. *arXiv preprint arXiv:1902.09599*, 2019.

[105] Roderick JA Little. A test of missing completely at random for multivariate data with missing values. *Journal of the American statistical Association*, 83(404):1198–1202, 1988.

[106] Roderick JA Little. A class of pattern-mixture models for normal incomplete data. *Biometrika*, 81(3):471–483, 1994.

[107] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[108] Jinfei Liu, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jimeng Sun. Dealer: an end-to-end model marketplace with differential privacy. *Proceedings of the VLDB Endowment*, 14(6), 2021.

[109] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309, 2019.

[110] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. The shapley value of tuples in query answering. In *ICDT*, volume 155, pages 20:1–20:19, 2020.

[111] Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. Approximate denial constraints. *Proc. VLDB Endow.*, 13(10):1682–1695, 2020.

[112] Ester Livshits and Benny Kimelfeld. The shapley value of inconsistency measures for functional dependencies. *Log. Methods Comput. Sci.*, 18(2), 2022.

[113] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing optimal repairs for functional dependencies. *ACM Trans. Database Syst.*, 45(1):4:1–4:46, 2020.

[114] Ester Livshits, Rina Kochirgan, Segev Tsur, Ihab F. Ilyas, Benny Kimelfeld, and Sudeepa Roy. Properties of inconsistency measures for databases. In *SIGMOD*, pages 1182–1194, 2021.

[115] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. E2gan: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of the 28th international joint conference on artificial intelligence*, pages 3094–3100. AAAI Press, 2019.

[116] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, 2018.

[117] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. AIM: an adaptive and iterative mechanism for differentially private synthetic data. *CoRR*, abs/2201.12677, 2022.

[118] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *ICML*, volume 97, pages 4435–4444, 2019.

[119] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

[120] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul, editors, *SIGMOD*, pages 19–30. ACM, 2009.

[121] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[122] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[123] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530, 2019.

[124] Shubhankar Mohapatra, Amir Gilad, Benny Kimelfeld, and Xi He. https://github.com/mshubhankar/DPMeasures. Full paper and codebase.

[125] Shubhankar Mohapatra, Sajin Sasy, Xi He, Gautam Kamath, and Om Thakkar. https://github.com/mshubhankar/The-Role-of-Adaptive-Optimizers-for-Honest-Private-Hyperparameter-Selection. Full paper and codebase.

[126] Shubhankar Mohapatra, Sajin Sasy, Xi He, Gautam Kamath, and Om Thakkar. The role of adaptive optimizers for honest private hyperparameter selection. *arXiv preprint arXiv:2111.04906*, 2021.

[127] Shubhankar Mohapatra, Jianqiao Zong, Florian Kerschbaum, and Xi He. Differentially private data generation with missing data. https://github.com/mshubhankar/DP-DataGeneration-MissingData, 2023.

[128] John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

[129] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[130] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing data imputation using optimal transport. In *International Conference on Machine Learning*, pages 7130–7140. PMLR, 2020.

[131] Shinichi Nakagawa and Robert P Freckleton. Missing inaction: the dangers of ignoring missing data. *Trends in ecology & evolution*, 23(11):592–596, 2008.

[132] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 29th IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[133] Eric Schulte Nordholt. Imputation: methods, simulation experiments and practical examples. *International Statistical Review*, 66(2):157–180, 1998.

[134] AH Noruzman, NA Ghani, and NSA Zulkifli. Gretel. ai: Open-source artificial intelligence tool to generate new synthetic data.

[135] National Institute of Standards and Technology. Differential privacy synthetic data challenge, 2018.

[136] Department of Transportation. Flight. Research and Innovative Technology Administration, 2020. https://data.transportation.gov/d/7fzd-cqts.

[137] Oleh Onyshchak. Stock market dataset, 2020.

[138] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy, 2020.

[139] Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy. In *International Conference on Learning Representations*.

[140] Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy. *arXiv preprint arXiv:2110.03620*, 2021.

[141] Stuart L Pardau. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68, 2018.

[142] Francesco Parisi and John Grant. Inconsistency measures for relational databases. *arXiv preprint arXiv:1904.03403*, 2019.

[143] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, 2016.

[144] Shweta Patwa, Danyu Sun, Amir Gilad, Ashwin Machanavajjhala, and Sudeepa Roy. DP-PQD: privately detecting per-query gaps in synthetic data generated by black-box mechanisms. *Proc. VLDB Endow.*, 17(1):65–78, 2023.

[145] Eduardo H. M. Pena, Eduardo Cunha de Almeida, and Felix Naumann. Fast detection of denial constraint violations. *Proc. VLDB Endow.*, 15(4):859–871, 2021.

[146] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[147] Haoyue Ping, Julia Stoyanovich, and Bill Howe. Datasynthesizer: Privacy-preserving synthetic datasets. In *SSDBM*, pages 42:1–42:5. ACM, 2017.

[148] Eugenia Politou, Efthimios Alepis, and Constantinos Patsakis. Forgetting personal data and revoking consent under the gdpr: Challenges and proposed solutions. *Journal of cybersecurity*, 4(1):tyy001, 2018.

[149] CORGIS Dataset Project. Hospitals csv file. https://corgis-edu.github.io/corgis/csv/hospitals/.

[150] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *SIGMOD*, pages 1435–1446, 2014.

[151] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 1999.

[152] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031*, 2020.

[153] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[154] Jalan Rivers, Andrea Nelson, and Lauren Williams. Synthetic data generation with sdv.

[155] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[156] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[157] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98, 2011.

[158] Tom Sander, Yaodong Yu, Maziar Sanjabi, Alain Durmus, Yi Ma, Kamalika Chaudhuri, and Chuan Guo. Differentially private representation learning via image captioning. *arXiv preprint arXiv:2403.02506*, 2024.

[159] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, SP '17, pages 3–18, Washington, DC, USA, 2017. IEEE Computer Society.

[160] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, pages 2242–2251. IEEE Computer Society, 2017.

[161] James B Simon, Dhruva Karkada, Nikhil Ghosh, and Mikhail Belkin. More is better in modern machine learning: when infinite overparameterization is optimal and overfitting is obligatory. *arXiv preprint arXiv:2311.14646*, 2023.

[162] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *GlobalSIP*, pages 245–248, 2013.

[163] Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022.

[164] Peng Sun, Xu Chen, Guocheng Liao, and Jianwei Huang. A profit-maximizing model marketplace with differentially private federated learning. In *INFOCOM*, pages 1439–1448. IEEE, 2022.

[165] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in Neural Information Processing Systems*, 26:2004–2012, 2013.

[166] Uthaipon Tao Tantipongpipat, Chris Waites, Digvijay Boob, Amaresh Ankit Siva, and Rachel Cummings. Differentially private synthetic mixed-type data generation for unsupervised learning. In *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–9. IEEE, 2021.

[167] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. Computing local sensitivities of counting queries with joins. In *SIGMOD*, pages 479–494, 2020.

[168] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238*, 2021.

[169] Christine Task, Karan Bhagat, Streat Damon, and Gary Howarth. NIST Diverse Community Excerpts Data, December 2022.

[170] Matthias Thimm. On the compliance of rationality postulates for inconsistency measures: A more or less complete picture. *KI-Künstliche Intelligenz*, 31:31–39, 2017.

[171] Zhihua Tian, Jian Liu, Jingyu Li, Xinle Cao, Ruoxi Jia, and Kui Ren. Private data valuation and fair payment in data marketplaces. *CoRR*, abs/2210.08723, 2022.

[172] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[173] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: differentially private synthetic data and label generation. *CoRR*, abs/2001.09700, 2020.

[174] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[175] Florian Tramèr and Dan Boneh. Differentially private learning needs better features (or much more data). In *Proceedings of the 9th International Conference on Learning Representations*, ICLR '21, 2021.

[176] Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*, 2021.

[177] Stef Van Buuren. *Flexible imputation of missing data*. CRC press, 2018.

[178] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pages 5998–6008. Curran Associates, Inc., 2017.

[179] Vijay V Vazirani. Approximation algorithms. *Georgia Inst. Tech*, 1997.

[180] Christopher Waites. Pyvacy: Towards practical differential privacy for deep learning. *https://github.com/ChrisWaites/pyvacy*, 2019.

[181] Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In *NIPS*, pages 2451–2459, 2010.

[182] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private sql with bounded user contribution. *arXiv preprint arXiv:1909.01917*, 2019.

[183] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. Attention-based learning for missing data imputation in holoclean. In *MLSys*, 2020.

[184] Mingyan Xiao, Ming Li, and Jie Jennifer Zhang. Locally differentially private personal data markets using contextual dynamic pricing mechanism. *IEEE Trans. Dependable Secur. Comput.*, 20(6):5043–5055, 2023.

[185] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.*, 23(8):1200–1214, 2011.

[186] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *CoRR*, abs/1802.06739, 2018.

[187] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

[188] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

[189] Yungang Xu, Zhigang Zhang, Lei You, Jiajia Liu, Zhiwei Fan, and Xiaobo Zhou. scigans: single-cell rna-seq imputation using generative adversarial networks. *Nucleic acids research*, 48(15):e85–e85, 2020.

[190] Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pages 5689–5698. PMLR, 2018.

[191] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large scale private learning via low-rank reparametrization. *International Conference of Machine Learning (ICML) 2021*, 2021.

[192] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex. Differentially Private Model Publishing for Deep Learning. In *2019 IEEE Symposium on Security and Privacy (SP'19)*, 2019.

[193] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: private data release via bayesian networks. In *SIGMOD*, pages 1423–1434, 2014.

[194] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 731–745, 2015.

[195] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.

[196] Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD*, pages 155–170, 2016.

[197] Yunjia Zhang, Zhihan Guo, and Theodoros Rekatsinas. A statistical perspective on discovering functional dependencies in noisy data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 861–876, New York, NY, USA, 2020. Association for Computing Machinery.

[198] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven HOI, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *arXiv preprint arXiv:2010.05627*, 2020.

[199] Tianqing Zhu, Gang Li, Wanlei Zhou, and Philip S. Yu. Differentially private data publishing and analysis: A survey. *IEEE Trans. Knowl. Data Eng.*, 29(8):1619–1638, 2017.