Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Pengfei Tang Date

Deep Learning with Differential Privacy and Adversarial Robustness

By

Pengfei Tang
Doctor of Philosophy

Department of Computer Science

Li Xiong, Ph.D.
Advisor

Lance Waller, Ph.D.
Committee Member

Liang Zhao, Ph.D.
Committee Member

Ming Li, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D, MPH
Dean of the James T. Laney School of Graduate Studies

Date

Deep Learning with Differential Privacy and Adversarial Robustness

By

Pengfei Tang
B.A., Zhejiang University, Hangzhou, China, 2015

Advisor: Li Xiong, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Department of Computer Science
2021

Abstract

Deep Learning with Differential Privacy and Adversarial Robustness
By Pengfei Tang

Deep learning models have been increasingly powerful on different tasks, such as image classification and data synthesization. However, there are two major vulnerabilities existing: 1) privacy leakage of the training data through inference attacks, and 2) adversarial examples that are crafted to trick the classifier to misclassify. Differential privacy (DP) is a popular technique to prevent privacy leakage, which offers a provable guarantee on privacy of training data through randomized mechanisms such as gradient perturbation. For attacks of adversarial examples, there are two categories of defense: empirical and theoretical approaches. Adversarial training is one of the most popular empirical approaches, which injects adversarial examples with correct labels to the training dataset and renders the model robust through optimization. Certified robustness is a representative of theoretical approaches, which offers a theoretical guarantee to defend against adversarial examples through randomized mechanisms such as input perturbation. However, there are some limitations in existing works that reduce the effectiveness of these approaches. For DP, one challenge is the contradiction between a better utility performance and a certain level of privacy guarantee. For adversarial training, one challenge is that when the types of adversarial examples are limited, the model robustness is confined. For certified robustness, existing works fail to exploit the connection between input and gradient perturbation, which wastes a part of randomization during training.

To solve these limitations, 1) we propose a novel framework IGAMT for data synthesization. Compared with traditional frameworks, IGAMT adds less gradient perturbation to guarantee DP, but still keeps the complex architecture of generative models to achieve high utility performance. 2) We propose a distance constrained Adversarial Imitation Network (AIN) for generating adversarial examples. We prove that compared with traditional adversarial training, adversarial training with examples from AIN can achieve comparable or better model robustness. 3) We propose a new framework TransDenoiser to achieve both DP and certified robustness, which utilizes all randomization during training and saves the privacy budget for DP.

Deep Learning with Differential Privacy and Adversarial Robustness

By

Pengfei Tang
B.A., Zhejiang University, Hangzhou, China, 2015

Advisor: Li Xiong, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Department of Computer Science
2021

Acknowledgments

Thanks to my advisor, Dr. Xiong, who gives me enlightenment and support on researches, encourages me to explore and find the truth behind every challenge. Thanks to my family, whose love is the source of strength that I count on all of my days. Thanks to my grandpa and grandma, who gave me the infinite love during my childhood, supported me with endless encouragement and taught me the primary principle of being a good man.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Recent studies have shown that deep learning models are increasingly powerful for solving difficult tasks in the real-world, including image classification [55, 51], natural language processing [85] and data synthesization [20, 45]. This strong capability of deep learning models requires a large amount of training data, which brings two major vulnerabilities: 1) privacy leakage of the training data through inference attacks on trained models, and 2) adversarial examples that are designed by adding small perturbations to clean examples in order to trick the classifier to misclassify. Research has shown that inference attacks [32, 88, 77] on models can incur privacy leakage on the training dataset. Differential privacy (DP) [26, 25, 27] is a commonly used technique to prevent privacy violation, which offers a provable guarantee on privacy of training data through randomized mechanisms such as gradient perturbation [79, 76]. On the other hand, more and more adversarial attack algorithms [81, 15, 38] have been proposed to generate powerful adversarial examples that can incur models misclassify. Correspondingly, the defense approaches have also been developed to against these attacks, which can be classified into two categories: empirical and theoretical approaches. Adversarial training [64, 83] is one of the most popular empirical approaches, which injects adversarial examples with correct labels to the training dataset and renders the model robust through optimization. Certified robustness

[60, 56, 61, 21, 58, 75, 74] is a representative of theoretical approaches, which offers a theoretical guarantee to defend against adversarial examples through randomized mechanisms such as input perturbation.

Most existing works solve one of these two vulnerabilities, i.e., either employ differential privacy to deep learning models or achieve model robustness, while a few works achieve both goals independently [72, 71]. However, there are some limitations in existing works that reduce the effectiveness of these approaches. For DP, one challenge is the contradiction between a better utility performance and a certain level of privacy guarantee. The most popular technique to accomplish DP in deep learning models is the gradient perturbation [79, 76], which perturbs the gradients of parameters with Gaussian noise and commonly works with Moment accountant [2] to provide a tight privacy analysis. Although gradient perturbation based DP is a general framework appropriate for all kinds of deep learning models, it can degrade the utility performance when solving difficult tasks, such as data synthesization [20, 45], in which the model architecture is complicated and the parameter size is large. The reason behind this is intuitive, because the gradient perturbation introduces randomization during training and perturbs the optimization process. A more complicated architecture with larger parameter size requires more randomization, and the performance thus suffers.

For adversarial training [64], one potential issue is that the diversity and quality of adversarial examples augmented to the training dataset affect the robustness of the final model. Adversarial training is a training strategy that adds adversarial examples with correct label into the training dataset. This augmentation helps model learn the characteristics of adversarial examples and thus defend against attacks. It is obvious that adding more diverse adversarial examples into the training dataset can facilitate the model robustness. Ensemble adversarial training [83] proves this by augmenting training data with different adversarial examples crafted on other pre-trained models and getting better performance. However, different types of adversarial examples generated from other models are not al-

ways accessible because of time and device constraints. When the types of adversarial examples are limited, the model robustness is confined.

For certified robustness [60, 56, 61, 21, 58, 75, 74], one potential flaw is the waste of randomization during training and the failure of utilizing this randomization for DP. Although theoretically, certified robustness only needs perturbations at testing phase, all existing works require the randomization, e.g., input perturbation, during training for model utility and robustness performance. This randomization added for certified robustness could have been leveraged to provide a certain level of DP, but is ignored in existing works. We observe that some works [72, 71] have achieved both DP and certified robustness on deep learning models. However, these works independently accomplish two goals, i.e., utilize input perturbation for certified robustness and gradient perturbation for DP. They fail to exploit the connection between input and gradient perturbation, which wastes a part of randomization during training and incurs the degradation of utility performance.

1.1 Research Contributions

This thesis proposes a set of techniques to improve the privacy and robustness of deep learning models. 1) For DP, we propose a novel framework IGAMT to solve difficult tasks, i.e., temporal data synthesization. Compared with traditional frameworks, IGAMT introduces less randomization to guarantee DP, but still keeps the complex architecture of generative models to achieve high utility performance. 2) For adversarial training, we propose a distance constrained Adversarial Imitation Network (AIN) to generate adversarial examples of high quality and great diversity. We prove that compared with traditional adversarial training, adversarial training with examples from AIN can achieve comparable or better adversarial robustness. 3) For certified robustness, we propose a novel framework TransDenoiser to achieve both DP and certified robustness, which utilizes all randomization during training and saves the privacy budget for DP. The details are as follows:

Differentially Private Synthesization on Temporal Data (Chap. 4). We propose Imitative Generative Adversarial Mixed-embedding Transformer (IGAMT) to generate differentially private EHRs with sophisticated characteristics. IGAMT contains three generative adversarial networks (GANs) [37] and an autoencoder [41]. We claim that IGAMT is the first framework to generate differentially private EHRs of high quality with heterogeneous features, missing values and irregular measures. IGAMT solves challenges in existing works: 1) It leverages transformer [85] to capture both temporal and non-temporal features and leverages features embeddings to better fit non-temporal features. It also utilizes masks and time embedding to capture missing values and irregular measures. 2) It combines sequence-to-sequence autoencoder with transformer and GAN to better generate synthetic EHRs satisfying sophisticated characteristics. 3) It leverages a specific structure, imitator, to reduce the randomization brought by DP technique while still keeping a complex architecture and the same level of DP guarantee. This work is under review of ICLR 2022.

Contributions. Our key contributions are:

1. We propose IGAMT, the first generative model to generate differentially private EHRs with heterogeneous features, missing values and irregular measures.

2. We leverage sequence-to-sequence transformer incorporated with missing value masks, time embedding and non-temporal embedding to learn the sophisticated characteristics of EHRs and generate synthetic data.

3. We add an imitator to IGAMT, which can imitate the behaviors of the decoder of IGAMT and learn characteristics from different sources of EHRs. Applying gradient perturbation to imitator based generative model reduces DP randomization , which thus improves the synthesization performance while preserving the same level of DP.

4. We build several baseline models and ablation cases, and conduct a large volume of experiments on two datasets to prove that IGAMT with differential privacy can

achieve best and state-of-art performance among these models.

Distance Constrained Adversarial Imitation Networks for Adversarial Training (Chap. 5). We propose a distance constrained Adversarial Imitation Network (AIN) for generating both targeted and untargeted adversarial examples. AIN combines the benefit of both generative models and the traditional optimization based approaches, achieving fast generation and comparable performance with optimization based methods. This work has been published in TDSC 2021. Our main contributions are as follows:

1. We propose a distance constrained Adversarial Imitation Network (AIN) for generating both targeted and untargeted examples. It i) explicitly uses distance constraint to bound the perturbation scale and allows flexible tradeoff between perturbation scale and attack success rate, ii) imitates adversarial examples created by optimization based attacks in the learning process to achieve improved quality of the generated adversarial examples, iii) generates multiple adversarial examples given one clean example by injecting random noises into the central hidden layer of the autoencoder, and iv) leverages improved network architecture, e.g., self-attention [85], residual [63] and label conditioning [94, 101], to allow better quality of the generated examples.

2. We introduce a training framework that progressively and adaptively changes the distance constraint to train AIN in a stable and effective way. By starting from a loose constraint and progressively moving to a tighter constraint, the training framework facilitates the optimization and dramatically improves the attack performance.

3. We conduct experiments on several datasets and the results show that: i) AIN can provide high quality adversarial examples. Compared to existing state-of-the-art generative models like ATN and AdvGAN, AIN generates more optimal adversarial examples in terms of perturbation scale and attack success rate, and can generate large scale adversarial examples even when clean examples are limited. Compared

to gradient-based attacks like i-FGSM, AIN can generate better adversarial examples on MNIST and comparable examples on tiny-imagenet. AIN also achieves comparable or better classification accuracy on tiny-imagenet and better accuracy on MNIST in adversarial training compared with i-FGSM, thanks to the diversity and quality of its adversarial examples. ii) AIN can generate adversarial examples more efficiently. Compared with optimization based attack like C&W, AIN can generate comparable adversarial examples in terms of attack success rate but at a significantly faster speed.

Achieving both Differential Privacy and Certified Robustness for Pre-trained Classifiers (Chap. 6). We propose a novel framework TransDenoiser to simultaneously achieve certified robustness and DP for models with pre-trained classifiers. TransDenoiser has a similar architecture as [74] by adding a denoiser before pre-trained classifier. Compared with [74], TransDenoiser can provide similar level of certified robustness without retraining the pre-trained model, as well as guarantee DP for the training data. Compared with existing works that achieve both certified robustness and DP including SecureSGD [72] and StoBatch [71], TransDenoiser 1) provides a tigher guarantee of DP by utilizing all the randomization during training including input and gradient perturbations, and 2) achieves more effective certified robustness by leveraging randomized smoothing on the input instead of noisy layers in the model. This work is under review of TDSC 2022.

Contributions. Our key contributions are:

1. We propose a novel framework TransDenoiser that trains a denoiser through both input and gradient perturbation for achieving DP and certified robustness simultaneously on deep learning models with pre-trained classifiers. The input perturbation for achieving certified robustness is utilized to achieve partial DP and additional gradient perturbation is used as necessary for the overall DP, ensuring an enhanced privacy and utility performance.

2. We present an analytical tool that leverages Taylor expansion to transform input per-

turbation into gradient perturbation so that it can be quantified and composed with the explicit gradient perturbation for the DP guarantee. We propose a Multivariate Gaussian Mechanism (**MGM**) to analyze DP of the multivariate Gaussian perturbation and prove that **MGM** is a generalization of Heterogeneous Gaussian Mechanism [72].

3. Observing that the transformed gradient perturbation itself cannot satisfy the DP guarantee requirement in some scenario, we add additional perturbation following isotropic Gaussian distribution to the gradient, and propose Mixed Multivariate Gaussian Analysis (**MMGA**) to analyze the DP guarantee provided by transformed gradient perturbation and additional gradient perturbation. We also prove that **MMGA** can work with moments accountant [2] to provide a tight bound on the privacy cost.

4. We conduct extensive experiments on several benchmark datasets which demonstrate that TransDenoiser can 1) provide a significantly tighter bound on privacy cost with same utility performance, and 2) achieve similar level of certified robustness as other state-of-the-art works.

1.2 Organization

The remainder of this thesis is organized as follows. In Section 2, we give a brief overview of the related works. In Section 3, we provide some preliminaries on EHR, DP and certified robustness. Section 4, Section 5 and Section 6 introduce our three works. Section **??** concludes research progresses.

Chapter 2

Related Works

2.1 EHR Representation Learning

Several works focus on representation learning of EHR data by building specific neural networks to capture these characteristics. Neil et al. [68] propose a novel recurrent network, Phased-LSTM, to capture irregular measures of temporal data. Bang et al. [9] then leverages Phased-LSTM to fit EHRs with missing values and irregular measures. BERT model [23] of NLP area is also adapted to EHRs. Alsentzer et al. [4] propose ClinicalBERT to fit EHR text data. However, the disadvantage of ClinicalBERT is that it can only works on textual words, but cannot process numerical values of EHRs. Li et al. [62] propose an improved BERT model, BEHRT, for clinical EHRs with textual data. BHERT takes temporal features as embedding, and takes age and time steps of records as additional embedding to incorporate these characteristics into training process. Similar to ClinicalBERT, BHERT can neither handle numerical values, nor generate synthetic EHRs.

2.2 EHR Synthesization

There are also several works focus on EHR synthesization. Choi et al. [20] propose medGAN to generate multi-label discrete records via a combination of autoencoder (AE)

[41] and generative adversarial network (GAN) [37]. However, medGAN only works on discrete features but misses the potential privacy leakage issue. Hyland et al. [45] combine conditional GAN [66] with recurrent network [44, 19] and propose recurrent conditional GAN (RCGAN), which can generate temporal medical features. However, RCGAN does not take non-temporal features, the missing values of medical data and privacy leakage issue into consideration. Xu et al. [93] leverage conditional GAN to build CTGAN for tabular medical data, whose characteristics are quite different from the EHRs listed before. Baowaly et al. [10] improve medGAN by replacing the GAN with more powerful variants, WGAN [5, 39] and boundary-seekinfg GAN (BGAN) [43]. The proposed models are named medWGAN and medBGAN correspondingly. Similar to medGAN, this paper does not take temporal features and privacy leakage into consideration. To obtain a privacy-preserving generative model, Beaulieu [13] apply differential privacy [27] into the training process of the discriminator of AC-GAN [69]. Thus attackers cannot infer private information from the generated data. However, this work does not take the temporal features and missing values into consideration. Chin et al. [18] propose a DP GAN to generate heterogeneous EHRs. Heterogenous EHRs contains non-temporal features (e.g, Bernoulli features, categocial features and numerical features) and missing values, which is more complex than the dataset used in [20, 13, 10]. However, temporal features are still missed in this work. Lee et al. [57] propose dual adversarial autoencoder (DAAE) to generate temporal EHRs and employ differential privacy during training to prevent privacy leakage. DAAE is is the existing state-of-the-art generative model for EHRs, which leverages autoencoder, WGAN and recurrent networks to generate synthetic temporal EHRs with privacy guarantee, but is incapable of capturing non-temporal features, missing values and irregular measures. We adopt the similar architectures of DAAE, but replace the recurrent unit with more advanced network, i.e., transformer [85], to generate EHRs satisfying all 5 characteristics with better performance.

2.3 Differential Privacy

Gradient Perturbation. Gradient perturbation is a widely used technique that injects perturbation to the gradient of each parameter to guarantee DP for deep learning models. Song et al. [79] first propose the gradient perturbation method by injecting perturbation to the gradients during parameter updates with stocastic gradient descent (SGD). Bassily et al. [12] improve the gradient perturbation by leveraging privacy amplification via sampling [14] (Lemma II.2 in [12]) and strong composition [28] (Lemma II.3 in [12]) to achieve a tighter bound. Abadi et al. [2] make further improvement by proposing a novel privacy composition tool: moments accountant, which can compute the overall privacy cost during training and achieve a tighter bound. Shokri et al. [76] propose the gradient perturbation method under the distributed learning scenario. Wang et al. [86] replace SGD optimizer used in previous work with stochastic variance-reduced gradient (SVRG) [90] to achieve a faster optimization. However, it requires the loss function l to be convex, G-Lipschitz and β-smooth. Lee et al. [59] and Yu et al. [96] improve the gradient perturbation method by dynamically allocating the privacy budget per iteration and leverage zero-concentrated DP (zCDP) [59] to analyze the privacy cost.

Input Perturbation. Input perturbation is a technique that adds noise to the original training data to achieve DP models. Fukuchi et al. [34] first attempted to use Taylor expansion to transform input perturbation into gradient perturbation. Although input perturbation framework theoretically guarantees that model trained with perturbed inputs is DP, this work imposes several constraints on the loss function, which cannot be practically applied with deep learning systems. Kang et al. [48] propose an input perturbation that generalizes the constraints on the loss function to less strict conditions. They also take a further step by finding that different training data will affect the model in different ways [47]. However, this work requires a pre-trained model that should also be DP, which also requires privacy budget. In summary, all the above works impose strict constraints on the loss function to

analyze DP for input perturbation. These constraints can not be satisfied by typical deep learning models.

Matrix-valued Gaussian Mechanisms. Chanyaswad et al. [17] propose matrix-valued Gaussian (MVG) Mechanism to guarantee DP of matrix-valued query. MVG can be regarded as a general form of our proposed Multuvariate Gaussian Mechanism (**MGM**) However, there are several differences between our work and [17]. [17] focuses on DP guarantee of matrix-valued query, while our work focuses on deep learning models. These two settings are quite different: in deep learning setting, the perturbation is added to each iteration during training, while in query setting, the perturbation is added only for a few times. Compared with MVG, DP analysis of **MGM** in our work has less time cost, because it only requires the calculation of minimum singular value, while MVG requires all singular values and harmonic numbers.In addition, **MGM** is a theoretical mechanism in our work. We also propose Mixed multivariate Gaussian Analysis (**MMGA**) to analyze the DP guarantee for empirical deep learning training algorithms, which has not been introduced in other works. Based on MVG, Yang et al. [95] propose two specific types of Matrix Gaussian Mechanisms and prove that the utility performance of these mechanisms is better than MVG. However, these two mechanisms can not be applied to our framework. Because the transformed gradient perturbation in our work follows a multivariate Gaussian distribution that is neither of these two cases.

2.3.1 Privacy-preserving Generative Models

The generative models introduced in Section 2.2 are built for EHRs synthesization. There are some generative models that are privacy-preserving but not for EHRs. Xie et al. [91] propose DP-GAN which adds perturbation to the gradient of discriminator and uses moment accountant [2] to analyze the DP guarantee. This paper claims that the since the discriminator is guaranteed as DP, the generator is correspondingly DP according to the post-processing property [27]. Zhang et al. [100] improve the DP-GAN of [91] by us-

ing the improved WGAN [39] and employing multi-fold optimization strategies, which include weight clustering, adaptive clipping and warm starting. Similarly, Xu et al. [92] builds GANobfuscator based on the improved WGAN and adaptive clipping as in [100]. Frigerio et al. [33] propose a general DP GAN framework that can generate different types of data with privacy preservation, from time-series to continuous data, and discrete data. Torkzadehmahani et al. [82] propose DP-CGAN to generate synthetic data with corresponding labels. The training process separates the discriminator loss between real data and synthetic data to preserve information from discriminator loss on real data. DP-CGAN also proposes RDP accountant to provide a tighter bound on DP guarantee compared to moments accountant [2]. Augenstein et al. [6] incorporate differentially private generative models with federated learning, where data is distributed among users and the central server. This federated DP-GAN aims to identify the potential bugs in the data preprocessing, and can only guarantee user-level differential privacy. Fan et al. [29] conduct a survey to investigate and analyze the differentially private generative models. They compare the detailed differences among different models w.r.t. application domains, training procedure and evaluation metrics.

2.4 Existing Adversarial Attacks

Optimization based attacks. L-BFGS method [81] adds the distance constraint as a penalty term into the objective function and line-searches its coefficient to craft adversarial examples. Fast Gradient Sign Method [38] calculates the gradients on $\mathbf{x}$ in a single step to formulate the perturbation, which is fast to execute but sacrifices perturbation scale. Iterative FGSM (i-FGSM) [52] extends FGSM by iteratively adding gradients onto the image, which enhances the quality with slightly more computation cost. JSMA [70] uses the gradients to first compute a saliency map and then finds the most important pixel to increase the likelihood of the targeted class after modification. As JSMA calculates the Jacobian

matrix of given inputs, it is relatively slow compared to other methods. DeepFool [67] iteratively adds gradient based perturbations to cause the change of the classification. Carlini and Wagner (C&W) [16] leverages the minimization of a non-linear mapped perturbation to achieve much smaller perturbation scale compared with all other attacks, sacrificing the computation speed by three orders of magnitude. Decoupled Direction and Norm(DDN) [73] improves the time efficiency of C&W while still keeping similar high attack success rate by decoupling the direction and the norm of the adversarial perturbation. However, it is still slower than i-FGSM and AIN. Boundary projection (BP) [98] is more of an iterative method that uses two stages to generate adversarial examples: 1) leveraging the gradient of loss to find an adversarial example, and 2) lowering the distortion while keeping the image adversarial. This two-stage strategy can improve the attack success rate (compared to i-FGSM), but impairs the time efficiency.

Model training based attacks. Instead of directly solving the optimization problem for generating adversarial examples, several works proposed model training based methods that learn to generate adversarial examples. Adversarial Transformation Network (ATN) [8] presents an autoencoder based network which can generate a targeted adversarial example $\mathbf{x}'$ given a clean example $\mathbf{x}$. A reranking function is designed to modify the softmax probability of the target classifier given the clean example $\mathbf{x}$ and maximize the probability value at the position of targeted class $\mathbf{t}$. The autoencoder network is trained to minimize $\|\mathbf{x} - \mathbf{x}'\|_2$, and force the output of the target classifier given the adversarial example to be similar to the reranked probability of the clean example. Other works [89, 80, 87] leverage GANs to generate adversarial examples. Generators proposed in [80, 87] can generate adversarial examples from random distributions. However, since the adversarial examples do not have direct correspondence with clean examples, the perturbation can be significantly large and visually noticeable compared with ATN and optimization based methods. [89] proposes AdvGAN with an autoencoder based generator which takes a clean example $\mathbf{x}$ and outputs the perturbation.

2.5 Certified Robustness

PixelDP. Lecuyer et al. [56] propose PixelDP to achieve certified robustness by considering an input image as a database in DP parlance and each pixel of the image as each record in DP. PixelDP shows that adding a randomization layer in the model to preserve DP on image pixels guarantees certified robustness of the model against adversarial examples.

Randomized Smoothing. Randomized smoothing is another technique that adds random noise to the input for achieving certified robustness and has been shown to outperform PixelDP with tighter robustness guarantee. Li et al. [61] derive a certified bound for robustness to adversarial examples using R$\acute{e}$nyi Divergence [35] by adding additive random noise to the input. Cohen et al. [21] leveraged Neyman-Person lemma to analyze the correlation between the highest scored class and the second highest class. Compared to the previous work, they provide a tight certified robustness guarantee for the model. All above-mentioned work are certified within an L_2 radius which means that the adversary cannot alter the prediction within a L_2 unit ball. Lee et al. [58] provide certified robustness for discrete cases where the adversary is L_0 bounded (the number of pixel changes in a figure). Salman et al. [75] further employ adversarial training to improve the certified robustness of models. To provide more efficient certified robustness, Salman et al. [74] also propose to obtain a certified robust classifier from a fixed pre-trained model. Randomized Smoothing is then applied during testing and provides certified robustness without retraining the pre-trained model. In this paper, we adopt a similar setting as [74] but focus on achieving both certified robustness and DP at the same time via randomized smoothing (input perturbation).

2.6 Differential Privacy with Certified Robustness

There are few works on simultaneously achieving both DP and certified robustness. Phan et al. [72] first attempt to propose a framework called Secure-SGD to simultaneously achieve

certified robustness and differential privacy. They use a PixelDP [56] based approach to achieve certified robustness and propose a Heterogeneous Gaussian Mechanism (HGM) to improve the performance of certified robustness. To achieve DP, they use gradient perturbation and propose a Heterogeneous Gaussian Mechanism (HGM) by adding heterogeneous Gaussian noise instead of element-wise Gaussian noise. There are two major limitations of this work: 1) The random perturbation used to achieve certified robustness during training should have contributed a certain degree of randomness to preserve DP but is ignored. Instead, additional Gaussian perturbation is added onto the gradient to achieve DP. 2) The robustness bound provided by PixelDP is loose compared with other randomized smoothing [61, 21, 58] based approaches for certified robustness.

Another work from Phan et al. [71] developed StoBatch algorithm to guarantee DP and certified robustness. It first leverages Autoencoder (AE) [42] and functional mechanism (objective perturbation) [99] to reconstruct input examples with DP. Then, these reconstructed DP data is used to train a deep neural network for classification. To make the neural network robust and DP, they apply adversarial training [83] and functional mechanism as mentioned earlier during training. Although this framework achieves better performance compared to their previous work [72], there are still two limitations: 1) The approach is only applicable to simple architectures of AE due to the constraints of function mechanism and hence cannot learn complicated representation on high-dimensional or complex data. 2) Similar to Secure-SGD [72], the robustness bound provided by PixelDP is loose.

Chapter 3

Preliminaries

3.1 Differential Privacy

Differential Privacy (DP) [25, 26, 27] is a theoretical privacy framework for aggregate data analysis. It ensures the output distributions of a randomized algorithm are indistinguishable with a certain probability when running on two neighboring datasets differing in one record or bounded by a distance metric.

Definition 1. *((ϵ, δ)-Differential Privacy) A randomized mechanism $\mathcal{M} : \mathbf{D} \to \mathbf{R}$ with domain $\mathbf{D}$ and range $\mathbf{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $\mathcal{D}, \mathcal{D}' \in \mathbf{D}$ and for any subset of outputs $\mathbf{S} \subseteq \mathbf{R}$ it holds that*

$$Pr(\mathcal{M}(\mathcal{D}) \in \mathbf{S}) \leq e^{\epsilon} Pr(\mathcal{M}(\mathcal{D}') \in \mathbf{S}) + \delta,$$

where ϵ is the privacy budget and δ is the probability that privacy is broken.

The common mechanism to achieve (ϵ, δ)-DP is Gaussian Mechanism that adds calibrated noise to the output.

Theorem 1. *Gaussian Mechanism [27]. Let $\mathcal{G} : \mathbb{R}^v \to \mathbb{R}^w$ be an arbitrary w-dimensional function, and its sensitivity $\Delta_{\mathcal{G}} = \max_{\mathcal{D},\mathcal{D}'} \|\mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')\|_2$. A Gaussian Mechanism $\mathcal{M}$*

with σ adds element-wise noise $\mathcal{N}(0, \sigma^2)$ to the output. The mechanism $\mathcal{M}$ is (ϵ, δ)-DP, with

$$\epsilon \in (0, 1],\ \sigma \geq \sqrt{2\ln(1.25/\delta)}\Delta_{\mathcal{G}}/\epsilon.$$

Gaussian mechanism also works for deep learning models. Depending on where to inject the perturbations, we mainly introduce input perturbation and gradient perturbation that will be used in this paper.

Input Perturbation. Input perturbation [34, 47, 48] directly adds calibrated noise to input data to achieve DP. As a result, the objective function is generalized as,

$$\mathcal{J}_{priv}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l((\hat{\mathbf{x}}_{(i)}, \hat{y}_{(i)}), \theta),$$
$$\theta_{priv} = \operatorname{argmin} \mathcal{J}_{priv}(\theta),$$

where θ denotes the model parameters, $\mathcal{J}_{priv}$ denotes the perturbed objective function due to the perturbed input, θ_{priv} denotes the parameters of the final model, N is the size of the training dataset, $(\hat{\mathbf{x}}_{(i)}, \hat{y}_{(i)})$ denote the perturbed data and l is the loss function.

Gradient Perturbation. Gradient perturbation [79, 12, 76, 2, 3, 86, 59, 96] injects calibrated noise to the gradient during training with the following objective function and update steps.

$$\mathcal{J}(\theta_t) = \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{x}_i, y_i, \theta_t),$$
$$\theta_{t+1} = \theta_t - \eta(\nabla\mathcal{J}(\theta_t) + \mathbf{p}),$$

where θ_t denotes the parameter at training step t, $\nabla\mathcal{J}(\theta_t)$ denotes the gradient, and $\mathbf{p}$ denotes the gradient perturbation. The gradient $\nabla\mathcal{J}(\theta_t)$ is bounded by the clipping norm or constrained by Lipschitz continuity of loss function l.

3.2 Certified Robustness

Adversarial Examples. Adversarial examples are designed by adding small perturbations to clean examples, which are imperceptible to human eyes but can easily fool a deep learning model to produce incorrect output.

Generating an adversarial example $\mathbf{x}'$ can be expressed as a constrained optimization problem. For untargeted adversarial examples, it can be expressed as: given a clean input $\mathbf{x}$, its label y, and a classifier f, minimize $L(\mathbf{x}, \mathbf{x}')$, such that the prediction score at label y is not the maximum among all labels: $f_y(\mathbf{x}') \neq \max_{j=1,..,c} f_j(\mathbf{x}')$, where L is a distance metric, such as l_2 or l_∞, and c is the number of classes. The quality of adversarial examples is measured by both the perturbation scale, i.e. $L(\mathbf{x}, \mathbf{x}')$, and the misclassfication rate or attack success rate.

Certified Robustness. Certified robustness is a principled technique to defend against adversarial examples. A classifier f is certified robust if for any input $\mathbf{x} \in \mathcal{R}^v$, its prediction $\max_{j=1,...,c} f_j(\mathbf{x})$ is constant within some set around $\mathbf{x}$, that is:

$$\forall \tau \in l_p(\kappa) : f_y(\mathbf{x} + \tau) > \max_{j \neq y} f_j(\mathbf{x} + \tau),$$

where y is the label of $\mathbf{x}$, and $l_p(\kappa)$ denotes $\forall \tau \in \mathcal{R}^v, \|\tau\|_p \leq \kappa$.

Randomized Smoothing. Randomized smoothing is the state-of-the-art approach to achieve certified robustness. Given a classifier f, the random smoothing technique converts f into a smooth classifier g, s.t., for input $\mathbf{x}$, $\tilde{f}$ returns

$$\tilde{f}(\mathbf{x}) = \underset{j=1,...,c}{\operatorname{argmax}} Pr[f(\mathbf{x} + \mathbf{b}) = j],$$

where $\mathbf{b} \sim \mathcal{N}(0, \sigma^2 I)$.

Randomized smoothing calculates a certified radius κ, s.t.,

$$\forall \tau \in l_p(\kappa) : \tilde{f}(\mathbf{x} + \tau) = y,$$

where y is the label of $\mathbf{x}$, and $l_p(\kappa)$ denotes $\forall \tau \in \mathcal{R}^v, \|\tau\|_p \leq \kappa$.

Theoretically, randomized smoothing only works during the testing phase by injecting input perturbation to the testing samples. However, there is a common practice during the training phase, which perturbs the training samples with the input perturbation to improve the utility performance.

3.3 Model Architectures

VAE. Variational Autoencoder (VAE) [50] is a fundamental generative model, which includes an encoder and a decoder. The encoder transforms input into mean and standard deviation of Gaussian distribution, and the decoder takes samples from this distribution and generate synthetic data. The objective function of VAE contains two parts: a reconstruction loss and a Kullback–Leibler divergence loss that works on the distribution in VAE and the standard Gaussian distribution.

GAN. Generative Adversarial Network (GAN) [37] is one of the most powerful and popular generative model in deep learning, which contains a generator and a discriminator. During training, these two parts compete with each other and are optimized in an adversarial process. The discriminator aims to distinguish real data from synthetic data generated by generator, while the generator aims to generate synthetic data that cannot be distinguished.

WGAN. Wasserstein-GAN (WGAN) [5, 39] is an improved variant of GAN, which 1) creatively leverages Wasserstein distance as the measure of discrimination, and 2) applies gradient penalty to guarantee Lipschitz property. WGAN has been proved by a large volume of researches as a powerful variant of GAN.

AAE. Adversarial Autoencoder (AAE) [65] can be regarded as a variant, which leverages a discriminator to distinguish the central hidden state in AE and the samples from a standard Gaussian distribution. AAE utilizes this discrimination to replace the Kullback–Leibler divergence used in traditional VAE.

VAE-GAN. VAE-GAN [53] can also be regarded as a variant of VAE, which adds a discriminator to distinguish real data from reconstructed data and synthetic data. Besides the reconstruction loss and the Kullback–Leibler divergence loss, VAE-GAN requires discriminate loss during training.

DAAE. Dual Adversarial Autoencoder (DAAE) [57] combines AAE and VAE-GAN together by leveraging two discriminators to not only distinguish the central hidden state in AE, but also distinguish real data from reconstructed data and synthetic data. Its objective function contains the reconstruction loss and discrimination loss in two GANs.

Chapter 4

IGAMT: Synthesizing Temporal Electronic Health Records with Differential Privacy

4.1 Overview

Recent studies have shown that generating private electronic health records (EHRs) is increasingly meaningful for a wide range of clinical usage. However, there are still three major challenges: 1) The heterogeneous characteristic of containing both temporal and non-temporal features in EHRs makes it difficult to employ deep learning models. 2) Missing values and irregular measures of EHRs frequently encountered in practice pose further challenges. 3) The privacy issue of generated data can incur sensitive information leakage. In this paper, we propose a novel model, Imitative Generative Adversarial Mixed-embedding Transformer (IGAMT), to solve these three challenges. We leverage state-of-the-art architecture and networks to not only learn the characteristics of EHRs with heterogeneous features, missing values and irregular measures, but also generate synthetic data of high quality. We also propose a novel architecture to reduce the randomization brought by dif-

ferential privacy technique while still keeping the same level of privacy guarantee. We conduct a large volume of experiments to prove that IGAMT significantly outperforms baselines and we also conduct extensive experiments about ablation cases to prove the effectiveness of the architecture and techniques applied in IGAMT.

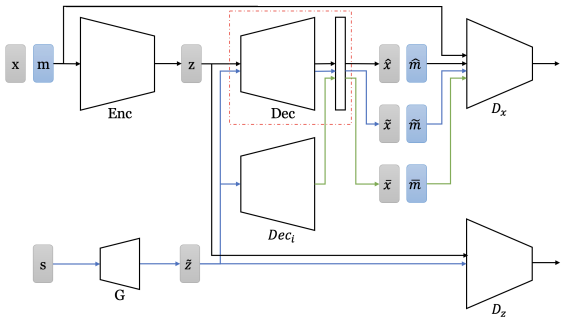


Figure 4.1: The framework of IGAMT, where Dec in red dashed box is virtually separated into two parts and shares the latter with Dec_i.

4.2 Imitative Generative Adversarial Mixed-embedding Transformer

As introduced in Section 1, generating differentially private EHRs with heterogeneous features, missing values and irregular measures has three challenges: 1) Representation learning of heterogeneous features, missing values and irregular measures. 2) Generation of synthetic EHR satisfying these characteristics. 3) Achieving differential privacy without compromising utility performance. In this section, we will 1) give an overview of our novel model IGAMT (Fig. 4.1), 2) present how IGAMT solves three challenges, and 3) introduce the training and synthesization of IGAMT.

4.2.1 Overview

As can be seen in Figure 4.1, IGAMT has a complicated architecture. It contains a seq2seq AE and three GANs: 1) Enc and Dec compose the seq2seq AE for representation learning. 2) Two GANs are crafted for Synthesization: G and D_z compose one GAN; G, Dec and

D_x compose another GAN. 3) G, Dec_i, the last layer of Dec and D_x compose the last GAN to synthesize EHRs while providing DP guarantee. Dec_i is the imitator we creatively proposed to reduce the randomization from gradient perturbation when achieving DP.

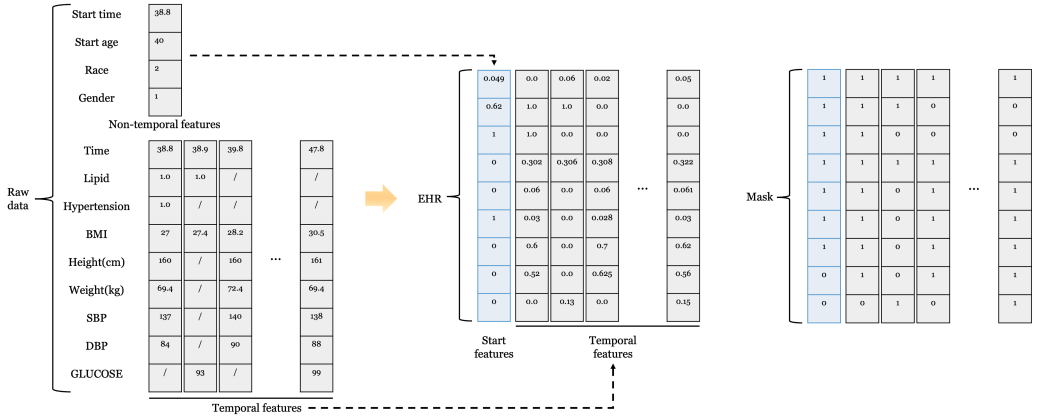


Figure 4.2: EHR raw data, preprocessing and missing value mask, where "/" denotes the missing value in features.

4.2.2 Representation Learning

In IGAMT, Enc and Dec compose the sequence-to-sequence autoencoder (seq2seq AE). Different from traditional autoencoder, seq2seq AE takes data with multiple time steps and generate data shifted with one time step. In IGAMT, non-temporal features are padded as the start features and concatenated to the start of input data. Thus seq2seq AE of IGAMT can learn characteristic of both non-temporal and temporal features during optimization. Moreover, seq2seq AE adopts the transformer [85] as the architecture and incorporates time embedding and non-temporal features embeddings into it, which further improve its capability of capturing heterogeneous characteristics. The detailed illustration of seq2seq AE can be found in Figure 1 of Appendix .1.

To capture missing values in EHRs, masks (Fig. 4.2) are crafted to element-wisely mark-off the missing value positions. Seq2seq AE of IGAMT will take both EHR data $\mathbf{x}$ and its mask $\mathbf{m}$ as the input, and generate corresponding synthetic data and mask. To capture irregular measures, time steps are extracted from EHRs and transformed into em-

bedding vectors. These embbeddings are then applied to the hidden states during training. Other non-temporal features like gender and race are also transformed into embedding vectors respectively and are broadcasted to all time steps before applying to hidden states. Time embedding renders IGAMT capturing irregular measures, and non-temporal features embeddings renders it further learning from non-temporal features. The details of embeddings can be found in Figure 4.3.

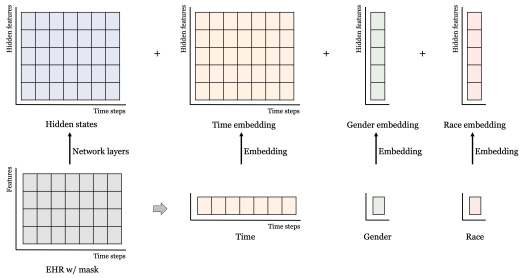


Figure 4.3: An illustration of time embedding and non-temporal feature embeddings, where "+" denotes the element-wise addition.

Losses on masked data. IGAMT takes both EHRs and missing value masks as input during training. Loss functions are thus designed on the masked data that element-wisely multiply EHR with its corresponding mask. We claim that losses on masked data can render IGAMT capturing characteristics related to missing values. In remains of Section 6.2, the element-wise multiplication is represented by "$\times$" and the masked data is represented by "data $\times$ mask".

Loss for representation learning. Sequence-to-sequence transformer autoencoder with specific embeddings in IGAMT is leveraged to capture the characteristics related to heterogeneous features, missing values and irregular measures. This autoencoder takes $\mathbf{x}$ and its mask $\mathbf{m}$ as the input and generate synthetic data $\hat{\mathbf{x}}$ and mask $\hat{\mathbf{m}}$, its reconstruction loss is either the cross-entropy loss (xEntropy):

$$l_{recon} = \text{xEntropy}(\mathbf{x} \times \mathbf{m},\ \hat{\mathbf{x}} \times \hat{\mathbf{m}}) + \text{xEntropy}(\mathbf{m},\ \hat{\mathbf{m}}),$$

or the mean square error (MSE):

$$l_{recon} = \text{MSE}(\mathbf{x} \times \mathbf{m},\ \hat{\mathbf{x}} \times \hat{\mathbf{m}}) + \text{MSE}(\mathbf{m}, \hat{\mathbf{m}}).$$

In our work, we employ the xEntropy as reconnstruction loss.

4.2.3 Generation

In IGAMT, GANs collaborates with seq2seq AE to generate synthetic EHRs with aforementioned sophisticated characteristics. The Generator G of GAN takes the random vectors $\mathbf{s}$ sampled from standard Gaussian distribution as the input and generate synthetic hidden state $\tilde{\mathbf{z}}$. $\tilde{\mathbf{z}}$ will be fed into Dec of seq2seq AE to generate corresponding synthetic data $\tilde{\mathbf{x}}$ and mask $\tilde{\mathbf{m}}$. Because during the reconstruction process of seq2seq AE, the hidden state $\mathbf{z}$ from Enc will be optimized to contain compressed information of EHRs. To improve the synthesization capability of generator, we leverage a discriminator D_z to discriminate $\mathbf{z}$ from $\tilde{\mathbf{z}}$. This discriminator D_z and the generator G compose a GAN in IGAMT.

Besides $\tilde{\mathbf{x}}$ and mask $\tilde{\mathbf{m}}$, another source of synthetic data is from the novel structure of IGAMT imitator Dec_i, which has the same architecture as Dec except that Dec_i excludes the last layer of Dec. As shown in Figure 4.1, Dec is virtually separated into two parts: last layer and all other layers. Dec_i takes $\tilde{\mathbf{z}}$ from G as input, and the output of Dec_i will firstly go through the last layer of Dec and then generate corresponding synthetic data $\bar{\mathbf{x}}$ and mask $\bar{\mathbf{m}}$.

To discriminate origin data and mask from others, discriminator D_x is crafted to take four different sources of data and masks as input: 1) the origin data $\mathbf{x}$ and its mask $\mathbf{m}$, 2) the reconstructed data $\hat{\mathbf{x}}$ and mask $\hat{\mathbf{m}}$ from seq2seq AE, 3) the data and mask generated from $\tilde{\mathbf{z}}$, i.e., $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{m}}$, and 4) $\bar{\mathbf{x}}$ and $\bar{\mathbf{m}}$ from Dec_i. Therefore, discriminator D_x, Dec, and G compose one GAN in IGAMT, and G, Dec_i, the last layer of Dec works with D_x to compose the last GAN in IGAMT. We claim that we utilize WGAN in IGAMT and will

not differentiate between GAN and WGAN in the remaining of this paper. Any statement about GAN should be regraded as WGAN without explicit clarification.

Losses for generation. Dec_i and Dec are generators in two GANs resepectively, they share the last layer of Dec and the discriminator D_x. The discriminate loss for D_x is as follows:

$$\begin{aligned} l_{dx} =&(D_x(\hat{\mathbf{x}} \times \hat{\mathbf{m}}) + D_x(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}}) + D_x(\bar{\mathbf{x}} \times \bar{\mathbf{m}}))- \\ & 3D_x(\mathbf{x} \times \mathbf{m}), \end{aligned}$$

the discriminate loss for Dec is as follows:

$$l_{de} = -(D_x(\hat{\mathbf{x}} \times \hat{\mathbf{m}}) + D_x(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}})),$$

and the loss for imitator Dec_i is as follows:

$$l_{di} = -D_x(\bar{\mathbf{x}} \times \bar{\mathbf{m}}).$$

As shown in these definitions, l_{dx} encourages D_x to discriminate $\mathbf{x}$, $\mathbf{m}$ from all other synthetic data and masks; l_{de} and l_{di} respectively encourages Dec and Dec_i to generate synthetic data that are more close to real data. Because reconstruction loss l_{recon} renders autoencoder capturing characteristics of real EHRs, the reconstructed data $\hat{\mathbf{x}}$ and mask $\hat{\mathbf{m}}$ are more challenging to be distinguished by D_x, which, in certain degree, benefits the generation of other two sources of synthetic data and masks.

For the GAN consists of the encoder Enc, the generator G and discriminator D_z. In this GAN, Enc provides "real" hidden states $\mathbf{z}$, G synthesize "fake" states $\tilde{\mathbf{z}}$, and D_z aims to distinguish $\mathbf{z}$ from $\tilde{\mathbf{z}}$. Therefore, the discriminate loss for D_z is:

$$l_{dz} = D_z(\tilde{\mathbf{z}}) - D_z(\mathbf{z}),$$

the loss for Enc is:

$$l_{en} = D_z(\mathbf{z}),$$

and the loss for G is:

$$l_g = -D_z(\tilde{\mathbf{z}}).$$

Since the central hidden state $\mathbf{z}$ captures the compressed characteristics of EHR given the reconstruction optimization on autoencoder, $\tilde{\mathbf{z}}$ that is more similar to $\mathbf{z}$ can encourage Dec and Dec_i to generate synthetic data and masks that are more similar to $\mathbf{x}$ and $\mathbf{m}$.

4.2.4 Imitation

Directly applying DP technique to IGAMT without imitator Dec_i will bring overwhelmed randomization to training process. Because both the decoder Dec and the discriminator D_z access the origin data, in order to release a differentially private generative model, perturbations require to be added to the gradients of Dec and D_z. As a comparison, DP GANs only adds gradient perturbations to the discriminators. Since the parameter size of Dec is much larger than discriminators, directly achieving DP on Dec and D_z will introduce overwhelmed randomization and thus degrades the utility performance.

To reduce the DP randomization in training, we leverage the novel structure of IGAMT, imitator Dec_i, which works with G the last layer of Dec and D_x to compose an extra GAN in IGAMT. This GAN is optimized in an adversarial process and encourages Dec_i to imitate the behavior of Dec during training. Compared with Dec, Dec_i will not directly access origin data $\mathbf{x}$ and mask $\mathbf{m}$, and all indirect accesses come from back-propagation during training.

As shown in Figure 4.1, IGAMT has two configurations of generative models: 1) Dec based model, which includes generator G and decoder Dec, and 2) Dec_i based model, which includes D, Dec_i and last layer of Dec. In Dec based model, since Dec accesses origin data $\mathbf{x}$ and $\mathbf{m}$, perturbations require to be added to the gradients of Dec and G

to guarantee DP. While in Dec_i based model, since only the last layer of Dec, D_x and D_z access origin data and mask, gradient perturbations on these parts and post-processing property can guarantee Dec_i and G to be DP.

Comparing these two configurations, we observe that Dec_i based model can avoid the complex architecture of Dec when achieving DP, and thus reduces overwhelmed randomization introduced by only adding gradient perturbations to D_x, D_z and last layer of Dec. The only potential concern is whether Dec_i can generate synthetic data as "real" as Dec or even as the real data. We claim that by leveraging a complicated and well-designed objective function, Dec_i can imitate the behaviors of Dec and learn characteristics from real EHRs and other sources of synthetic EHRs.

Loss for imitation. The imitative loss is creatively proposed to encourage the imitation of Dec's behaviors and the learning of characteristics from different sources of EHRs. We claim that a well-learned Dec_i can not only synthesize EHRs of high quality, but also save privacy costs during training. The loss for imitator Dec_i is as follows:

$$\begin{aligned} l_{im} =& \mathrm{MSE}(\hat{\mathbf{x}} \times \hat{\mathbf{m}},\ \bar{\mathbf{x}} \times \bar{\mathbf{m}}) + \mathrm{MSE}(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}},\ \bar{\mathbf{x}} \times \bar{\mathbf{m}}) + \\ & \mathrm{MSE}(\mathbf{x} \times \mathbf{m},\ \bar{\mathbf{x}} \times \bar{\mathbf{m}}). \end{aligned}$$

As can be seen in this definition, l_{im} encourages Dec_i to generate $\bar{\mathbf{x}}$ and $\bar{\mathbf{m}}$ that close to real data and other two sources of synthetic data, which encourages Dec_i to imitate the behaviors of Dec, and learn characteristics from real EHRs and other sources of synthetic EHRs.

4.2.5 Training

In previous sub-sections, we have introduced different losses to solve different challenges. During training, these losses works with different parts of IGAMT. In summary, the loss to

Algorithm 1: Training of IGAMT.

Input: preprocessed training EHRs $\mathbf{x}$ and masks $\mathbf{m}$, total training epoch T, gradient perturbation scale σ, learning rate η, batch size B, discriminators update frequency base fb and frequency hit fh, gradient clipping norm C

t = 0;
initialize parameters of IGAMT;
while $t < T$ **do**
  get mini-batch EHRs $\mathbf{x}_{(t)}$ and masks $\mathbf{m}_{(t)}$;
  $\mathbf{z}_{(t)} = Enc(\mathbf{x}_{(t)}, \mathbf{m}_{(t)})$;
  $\hat{\mathbf{x}}_{(t)}, \hat{\mathbf{m}}_{(t)} = Dec(\mathbf{x}_{(t)}, \mathbf{m}_{(t)}, \mathbf{z}_{(t)})$;
  $\tilde{\mathbf{z}}_{(t)} = G(B)$;
  start features $\mathbf{s}_f$ are sampled, start masks $\mathbf{s}_m$ are crafted;
  $\hat{\mathbf{x}}_{(t)}, \hat{\mathbf{m}}_{(t)} = Dec^{syn}(\mathbf{s}_f, \mathbf{s}_m, \tilde{\mathbf{z}}_{(t)})$;
  $\bar{\mathbf{x}}_{(t)}, \bar{\mathbf{m}}_{(t)} = Imi(\mathbf{s}_f, \mathbf{s}_m, \tilde{\mathbf{z}}_{(t)})$;
  if *t % fb ¡ fh* **then**
    // Update D_x with perturbation
    $l_{dx} = (D_x(\hat{\mathbf{x}}_{(t)} \times \hat{\mathbf{m}}_{(t)}) + D_x(\tilde{\mathbf{x}}_{(t)} \times \tilde{\mathbf{m}}_{(t)}) + D_x(\bar{\mathbf{x}}_{(t)} \times \bar{\mathbf{m}})_{(t)}) - D_x(\mathbf{x}_{(t)} \times \mathbf{m}_{(t)})$;
    calculate the gradient: $grad_{dx,(t)} = \frac{1}{B}\nabla_{\theta_{dx,(t)}} l_{dx}$;
    clip the gradient: $grad_{dx,(t)} = grad_{dx,(t)}/max(1, \|grad_{dx,(t)}\|/C)$;
    update parameter of D_x: $\theta_{dx,(t+1)} = \theta_{dx,(t)} - \eta\,(grad_{dx,(t)} + \mathcal{N}(0, \sigma^2))$;
    // Update D_z with perturbation
    $l_{dz} = D_z(\tilde{\mathbf{z}}_{(t)}) - D_z(\mathbf{z}_{(t)})$;
    calculate the gradient: $grad_{dz,(t)} = \frac{1}{B}\nabla_{\theta_{dz,(t)}} l_{dz}$;
    clip the gradient: $grad_{dz,(t)} = grad_{dz,(t)}/max(1, \|grad_{dz,(t)}\|/C)$;
    update parameter of D_z: $\theta_{dz,(t+1)} = \theta_{dz,(t)} - \eta\,(grad_{dz,(t)} + \mathcal{N}(0, \sigma^2))$;
  end
  denote parameters of Dec excluding the last layer as θ_{prev};
  denote parameters of Dec's last layer as θ_{last};
  // Update Dec excluding the last layer
  $l_{dec} = \mathrm{xEntropy}(\mathbf{x}_{(t)} \times \mathbf{m}_{(t)},\ \hat{\mathbf{x}}_{(t)} \times \hat{\mathbf{m}}_{(t)}) + \mathrm{xEntropy}(\mathbf{m}_{(t)},\ \hat{\mathbf{m}}_{(t)}) - (\mathrm{D_x}(\hat{\mathbf{x}}_{(t)} \times \hat{\mathbf{m}}_{(t)}) + \mathrm{D_x}(\tilde{\mathbf{x}}_{(t)} \times \tilde{\mathbf{m}}_{(t)}))$;
  $\theta_{prev,(t+1)} = \theta_{prev,(t)} - \eta\ \frac{1}{B}\nabla_{\theta_{prev,(t)}} l_{dec}$;
  // Update Enc
  $l_{enc} = \mathrm{xEntropy}(\mathbf{x}_{(t)} \times \mathbf{m}_{(t)},\ \hat{\mathbf{x}}_{(t+1)} \times \hat{\mathbf{m}}_{(t)}) + \mathrm{xEntropy}(\mathbf{m}_{(t)},\ \hat{\mathbf{m}}_{(t)}) + \mathrm{D_z}(\mathbf{z}_{(t)})$;
  update parameter of Enc: $\theta_{enc,(t+1)} = \theta_{enc,(t)} - \eta\ \frac{1}{B}\nabla_{\theta_{enc,(t)}} l_{enc}$;
  // Update Dec's last layer with perturbation
  calculate the gradient: $grad_{last,(t)} = \frac{1}{B}\nabla_{\theta_{last,(t)}} l_{dec}$;
  clip the gradient: $grad_{last,(t)} = grad_{last,(t)}/max(1, \|grad_{last,(t)}\|/C)$;
  $\theta_{last,(t+1)} = \theta_{last,(t)} - \eta\,(grad_{last,(t)} + \mathcal{N}(0, \sigma^2))$;
  // Update Dec_i
  $l_{imi} = -D_x(\bar{\mathbf{x}}_{(t)} \times \bar{\mathbf{m}}_{(t)}) + \mathrm{MSE}(\hat{\mathbf{x}}_{(t)} \times \hat{\mathbf{m}}_{(t)},\ \bar{\mathbf{x}}_{(t)} \times \bar{\mathbf{m}}_{(t)}) + \mathrm{MSE}(\tilde{\mathbf{x}}_{(t)} \times \tilde{\mathbf{m}}_{(t)},\ \bar{\mathbf{x}}_{(t)} \times \bar{\mathbf{m}}_{(t)}) + \mathrm{MSE}(\mathbf{x}_{(t)} \times \mathbf{m}_{(t)},\ \bar{\mathbf{x}}_{(t)} \times \bar{\mathbf{m}}_{(t)})$;
  update parameter of Dec_i: $\theta_{imi,(t+1)} = \theta_{imi,(t)} - \eta\ \frac{1}{B}\nabla_{\theta_{imi,(t)}} l_{imi}$;
  // Update G
  $l_g = -D_z(\tilde{\mathbf{z}}_{(t)})$;
  update parameter of G: $\theta_{g,(t+1)} = \theta_{g,(t)} - \eta\ \frac{1}{B}\nabla_{\theta_{g,(t)}} l_g$;
end
$\theta_{(T)} := \{\theta_{dx,(T)}, \theta_{dz,(T)}, \theta_{prev,(T)}, \theta_{last,(T)}, \theta_{imi,(T)}, \theta_{g,(T)}\}$;
Output $\theta_{(T)}$ and compute overall privacy cost through moments accountant.

update D_x is

$$\begin{aligned} l_{dx} =&(D_x(\hat{\mathbf{x}} \times \hat{\mathbf{m}}) + D_x(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}}) + D_x(\bar{\mathbf{x}} \times \bar{\mathbf{m}}))- \\ & 3D_x(\mathbf{x} \times \mathbf{m}); \end{aligned} \tag{4.1}$$

the loss to update D_z is

$$l_{dz} = D_z(\tilde{\mathbf{z}}) - D_z(\mathbf{z}); \tag{4.2}$$

the loss to update Dec is

$$\begin{aligned} l_{dec} =&l_{recon} + l_{de} \\ =&\text{xEntropy}(\mathbf{x} \times \mathbf{m},\ \hat{\mathbf{x}} \times \hat{\mathbf{m}}) + \text{xEntropy}(\mathbf{m},\ \hat{\mathbf{m}})- \\ & (D_x(\hat{\mathbf{x}} \times \hat{\mathbf{m}}) + D_x(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}})); \end{aligned} \tag{4.3}$$

the loss to update Enc is,

$$\begin{aligned} l_{enc} =&l_{recon} + l_{en} \\ =&\text{xEntropy}(\mathbf{x} \times \mathbf{m},\ \hat{\mathbf{x}} \times \hat{\mathbf{m}}) + \text{xEntropy}(\mathbf{m},\ \hat{\mathbf{m}}) + \text{D}_\text{z}(\mathbf{z}), \end{aligned} \tag{4.4}$$

the loss to update Dec_i is,

$$\begin{aligned} l_{imi} =&l_{di} + l_{im} \\ =&- D_x(\bar{\mathbf{x}} \times \bar{\mathbf{m}}) + \text{MSE}(\hat{\mathbf{x}} \times \hat{\mathbf{m}},\ \bar{\mathbf{x}} \times \bar{\mathbf{m}})+ \\ & \text{MSE}(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}},\ \bar{\mathbf{x}} \times \bar{\mathbf{m}}) + \text{MSE}(\mathbf{x} \times \mathbf{m},\ \bar{\mathbf{x}} \times \bar{\mathbf{m}}); \end{aligned} \tag{4.5}$$

the loss to update G is

$$l_g = -D_z(\tilde{\mathbf{z}}). \tag{4.6}$$

With these objective functions, different parts of IGAMT can be optimized during training. Algorithm 1 shows the training process of IGAMT. In this algorithm, we choose xEntropy for l_{recon} and use subscript (t) to denote the data at iteration t. At the start of

training, Enc and Dec work as a autoencoder to reconstruct EHRs and masks: 1) Enc takes a batch of EHRs $\mathbf{x}_{(t)}$ and masks $\mathbf{m}_{(t)}$ as input and outputs central hidden states $\mathbf{z}_{(t)}$ (line 5). 2) Dec then generates synthetic $\hat{\mathbf{x}}_{(t)}$ and $\hat{\mathbf{m}}_{(t)}$ (line 6). Dec then works as a generator (line 9), which takes start features $\mathbf{s}_f$, start masks $\mathbf{s}_m$ and synhtetic central states $\tilde{\mathbf{z}}_{(t)}$ from G as inputs, and synthesize $\hat{\mathbf{x}}_{(t)}$ and $\hat{\mathbf{m}}_{(t)}$. Different from reconstruction process of Dec, in synthesization process, Dec firstly generates synthetic output at first time step, and then it repeats this synthesization by taking output generated at previous time step as input and generating output of current step. We use Dec^{syn} to distinguish the synthesization from reconstruction of Dec. The synthesization of Dec_i is similar to Dec^{syn}, which generates $\bar{\mathbf{x}}_{(t)}$ and $\bar{\mathbf{m}}_{(t)}$ (line 10).

The remains of iteration t (line 11 - 27) show the optimization of different parts in IGAMT, which can be roughly divided into two stages: stage of updating discriminators (line 11 - 16) and stage of updating generators (line 17 - 27). In the former stage, the frequency of updating discriminators can be controlled by hyper-parameters fb and fh. To guarantee DP, gradient perturbations are added when updating discriminators (line 13, 15) and last layer of Dec (line 23).

4.2.6 DP and Synthesization

After training, the differentially private generative model of IGAMT can be leveraged to synthesize EHRs. The total privacy budget can be calculated follows Theorem 1 in [2], i.e.,

Theorem 2. *Given training batch size B, the total training size N, the number of training steps T, there exists constants c_1 and c_2 for any $\epsilon < c_1(\frac{B}{N})^2T$, Algorithm 1 is (ϵ, δ)-DP for any $\delta > 0$, if we choose*

$$\sigma \geq c_2 \frac{B/N\sqrt{Tlog(1/\delta)}}{\epsilon}.$$

As can be seen in Figure 4.4, this generative model contains G, Dec_i and the last layer of Dec.

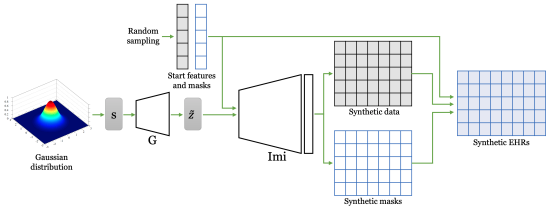


Figure 4.4: An illustration of synthesization.

The synthesization process can be divided into five stages. 1) Sampling of random states: random states $\mathbf{s}$ are sampled from a Gaussian distribution. 2) Generation of central hidden states: G takes $\mathbf{s}$ as the input and generate central hidden states $\tilde{\mathbf{z}}$. 3) Sampling of start features and masks: non-temporal features are randomly sampled from their corresponding ranges and assembled as the start features, their masks are crafted correspondingly. 4) Generation of temporal features and masks: Dec_i and the last layer takes features and masks generated at previous time steps as input and output features and masks for current time step. The features and masks at time step 0 are the start features and masks from stage 3. 5) Assembling: start and temporal features are concatenated to obtain complete EHRs, which are then marked with missing values by masks.

The synthetic EHRs from the last stage of this process includes heterogeneous features, missing values and irregular measures. Moreover, because the generative model is differentially private, these synthetic EHRs are correspondingly privacy preserving.

4.3 Experiments

In this section, we prove that IGAMT can generate DP synthetic EHRs 1) that are most visually-similar to real EHRs after PCA [31] and 2) that achieves highest similarity to real EHRs on downstream application such as clustering.

4.3.1 Configurations

Baseline and ablation studies. We slightly adapt the DAAE code to render it capable of taking and generating both EHR data and missing value mask. We also build three more baselines: GAN, VAE-GAN and AAE, and all of them take and generate EHR and mask. Besides these baselines, we build four ablation cases: $IGAMT\_w/o\_emb$, $IGAMT\_only\_t\_emb$, $IGAMT\_w/o\_imi$ and $IGAMT\_w/o\_imi\_var$. $IGAMT\_w/o\_emb$ is IGAMT without imitator and any embedding, $IGAMT\_only\_t\_emb$ is IGAMT without imitator but with only time embedding, $IGAMT\_w/o\_imi$ is IGAMT without imitator, and $IGAMT\_w/o\_imi\_var$ is a variant of $IGAMT\_w/o\_imi$, which enhances D_x with an extra discriminator D_m, which has been introduced in Appendix .1.

Implementation details. All models are implemented using Pytorch 1.8.1 and trained with a system equipped with Nvidia GeForce RTX 2070S. The learning rates for different parts of IGAMT are different: $5\mathrm{e}-4$ for Enc, Dec, G and Dec_i; $1\mathrm{e}-4$ for D_x and D_z. We leverage exponential learning rate decay with rates being 0.99 for all parts. For DP training, we set the ϵ as 1.5, δ as $1\mathrm{e}-5$, and use moment accountant to analyze DP cost. The detailed architectures for these models and the code are available here: `https://github.com/urihsam/Temporal-Synthesization`.

Evaluation metrics. Different evaluation metrics are used in out experiments. 1) PCA takes each record as a vector and reduces the dimension of data. It can visually display the result and show the difference between real and synthetic data. 2) Clustering with Minkowski distance. Clustering is a commonly-used application for unlabeled data, two similar datasets can have similar clustering results. We use Minkowski distance to measure this similarity of clustering centers from two datasets. The smaller minkowski distance is, the higher similarity is. 3) Clustering with cosine similarity. Cosine similarity is another similarity measurement. It measures the cosine of the angle between two vectors projected in a multi-dimensional space. We also apply it to two sets of clustering centers.

4.3.2 EHR and Preprocessing

EHRs. We use two EHR datasets in this work. One is Physionet MIMIC-IV-ED [36, 46]. The other one is from Emory Synergy project, which include "GLUCOSE", "DBP", "SBP", "BMI" and several other medical features. The left of the Figure 4.2 shows an example of EHR raw data, which includes two different types of features: non-temporal and temporal features, and missing values denoted as "/" exist among different features at different time steps. To better capture these features, EHR data is preprocessed before feeding into the model. After pre-processing, each record has 50 time steps with each time step having 9 and 10 features for MIMIC-IV-ED and Emory Synergy respectively. Because of the page limitation, we only show results of MIMIC-IV-ED in this section, the results of Emory Synergy will be shown in Appendix .2.

Non-temporal features preprocessing. For non-temporal features, we calculate the maximum and minimum values among all EHRs in the training dataset, and scale them to the range of $[0, 1]$. For discrete features, we transform them into one-hot vectors. For instance in Figure 4.2, start time 38.8 and age 40 are scaled to 0.049 and 0.63 respectively, and discrete features race 2 and gender 1 are transformed into $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ respectively. These transformed features along with zero padding are concatenated together to form the start features, which have the same size as the temporal features of each time step.

Temporal features preprocessing. EHRs contain several medical temporal features and an extra time features. For medical temporal features, we calculate the maximum and minimum values among all EHRs in the training data, and scale them to the range of $[0, 1]$. For time features, we first calculate the increment of two neighbouring time steps, and add 0 as the initial increment. Then we follow the similar process to scale these time increments to the range of $[0, 1]$. We also set the maximum number of time steps to 50, and pad some records having fewer time steps with 0.

Missing values and masks. As can be seen in Figure 4.2, EHR raw data contains missing values among different temporal features at different time steps. We first craft the masks to mark-off these missing values with 0 and mark on others with 1, and then replace the missing values with 0's among EHR features.

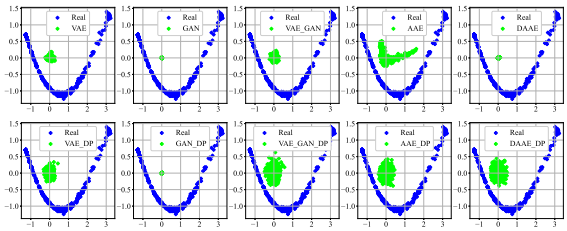


Figure 4.5: PCA for real EHRs and synthetic EHRs from baselines.

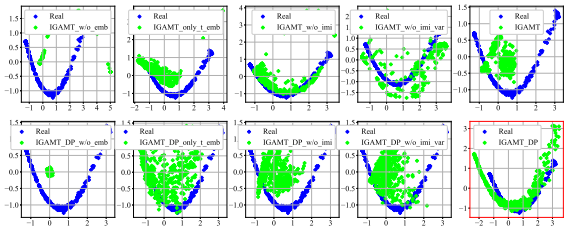


Figure 4.6: PCA for real EHRs and synthetic EHRs from ablation cases and IGAMT.

4.3.3 Data Visualization and Comparison

PCA Visualization

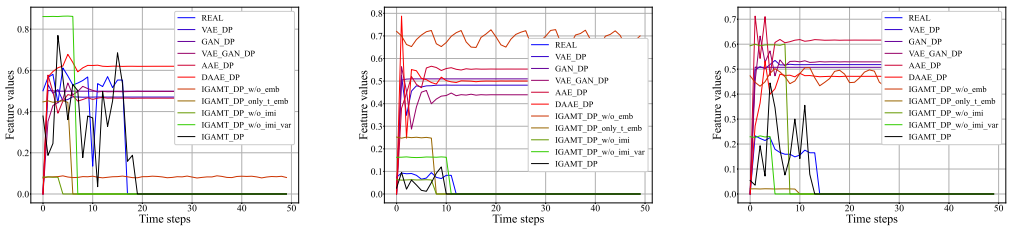
We use PCA to compress the real and synthetic data to 2 dimensional space and visually show the difference between real and synthetic EHRs. In Figure 4.5 and 4.6, the blue dots represents the compressed real EHRs and green dots represents the compressed synhtetic EHRs from different models.

Baselines. The first row of Figure 4.5 shows the results from five baselines; and second row shows the results from DP version of models corresponding to the first row. As can be observed, none of these models shows a similar pattern as the real EHRs. This indicates that 1) the architectures of these baseline models are not well-designed for the synthesization of temporal EHRs with missing values and irregular measures, and 2) DP technique applied during training can degrade the performance of models.

Table 4.1: Overall similarity with $\sigma = 1.0$.

Model	Minkowski Distance	Cosine Similarity
VAE_{DP}	5.907236	0.294431
GAN_{DP}	5.566364	-0.008107
VAE_GAN_{DP}	5.873504	0.015184
AAE_{DP}	5.555181	0.000039
$DAAE_{DP}$	5.578734	0.000111
$IGAMT\_w/o\_emb_{DP}$	5.547475	0.026667
$IGAMT\_only\_t\_emb_{DP}$	5.486770	0.171710
$IGAMT\_w/o\_imi_{DP}$	5.174218	0.239063
$IGAMT\_w/o\_imi\_var_{DP}$	5.111574	0.240675
$IGAMT_{DP}$	**3.974107**	**0.542739**

IGAMT and ablation cases. The first row of Figure 4.6 shows the results from four ablation models and IGAMT; and the second row also shows the DP version of these models. As can be seen, for non-DP version of models in the first row, DGAMT and $IGAMT\_w/o\_imi\_var$ show better results than others. However, for DP version in the second row, IGAMT provides the best result. For these four ablation cases and their corresponding DP versions, the model architecture becomes more and more complex from left to right. Therefore the results indicate that 1) for non-DP models, a more complex DGAT-based architecture can result in a better performance on synthesization, and 2) for DP models, a more complex architecture cannot improve the performance when the perturbation is directly applied on the gradient of generative parts. IGAMT leverages imitator to circumvent this challenge and adds perturbation on discriminate parts. As a result, IGAMT can not only have an architecture as complex as $IGAMT\_w/o\_imi\_var$, but also reduce the randomization introduced by DP and achieve best performance.



(a) Feature #0 (b) Feature #4 (c) Feature #7

Figure 4.7: Visualization of three features among all time steps

Table 4.2: Temporal feature similarity with $\sigma = 1.0$.

Model	Feature #0		Feature #4		Feature #7	
	Minko	Cosine	Minko	Cosine	Minko	Cosine
VAE_{DP}	1.098199	−0.04414	0.608728	0.404317	1.082422	−0.304460
GAN_{DP}	0.745575	−0.026782	0.778295	0.409500	1.461837	−0.348899
VAE_GAN_{DP}	1.018855	0.008946	0.470880	0.321513	0.923906	−0.213162
AAE_{DP}	0.801268	−0.000005	0.829536	0.069183	1.433803	−0.054064
$DAAE_{DP}$	0.724391	−0.012399	0.526601	0.342976	1.230185	−0.241429
$IGAMT\_w/o\_emb_{DP}$	0.799430	−0.07822	0.756629	−0.307154	1.407064	−0.070328
$IGAMT\_only\_t\_emb_{DP}$	1.061087	0.047748	1.335685	0.270278	1.285006	0.335499
$IGAMT\_w/o\_imi_{DP}$	1.502581	0.041217	0.482319	0.514935	1.130135	0.310963
$IGAMT\_w/o\_imi\_var_{DP}$	0.742870	0.008119	0.585838	0.485914	1.164596	**0.359482**
$IGAMT_{DP}$	**0.428964**	**0.38103**	**0.389896**	**0.580232**	**0.785058**	0.333837

Similarity Comparison

We calculate overall similarity between real clusters and synthetic clusters as follows, 1) pairing centers among real and synthetic clustering centers, 2) calculating pairwise similarity (e.g., Minkowski distances or cosine similarity) for these pairs, and 3) calculating the summation over all pairwise Minkowski distances or the average over all pairwise cosine similarity. As shown in Table 4.1, IGAMT outperforms all other models by achieving the smallest Minkowski distance and highest cosine similarity overall models.

4.3.4 Temporal Features Comparison

To provide a more detailed comparison of temporal features between real and synthetic EHRs, we pick three different temporal features, apply clustering on these features separately and calculate the similarity between clustering centers. We set the number of clusters as 3.

Table 4.2 shows the similarity results of different models, where "Feature #0", "Feature #4", "Feature #7" denote "time in year", "heart rate", "SBP" in temporal features respectively, "Minko" denotes Minkowski distance and "Cosine" denotes cosine similarity. As can be seen in this table, IGAMT almost dominate all other models w.r.t. both Minkowski distance and cosine similarity among three different features, esp. on feature #0 ("time

in year"), which proves that IGAMT not only captures the sophisticated characteristics of EHR features, but also preserves DP of training data with less randomization.

Besides the similarity comparison, we randomly samples 100 EHRs from real test data and synthetic data from DP models, and plot the average value of three selected features over 50 time steps. The results is as shown in Figure 4.7. Among different colored curves, the blue curve represents the real EHRs, black represents EHRs from IGAMTÀs can be seen in all three feature plots, black curves partially match the patterns of real features and outperform curves of all other models.

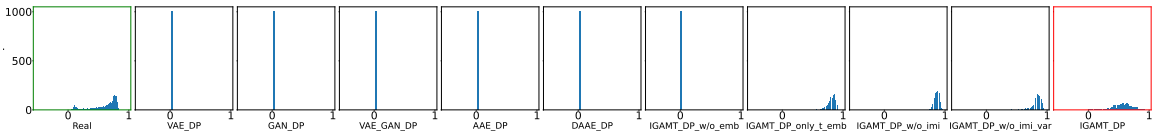


Figure 4.8: Missing values histogram of different EHRs

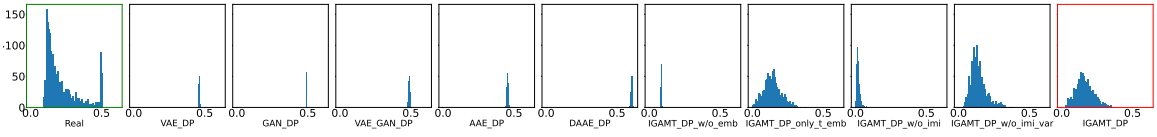


Figure 4.9: Irregular measures histogram of different EHRs

4.3.5 Missing Values and Irregular Measures

Missing values and Irregular Measures commonly exist in real EHRs. We incorporate missing value masks and time embedding into complicated architecture to capture these characteristics. To illustrate the statistics of missing values and irregular measures in EHRs, we 1) count the mark-off positions per feature in masks and plot the histogram of counted number averaged over features among 1000 samples, and 2) calculate the increased time between two neighbouring time steps and plot the histogram of increased time averaged over time steps among 1000 samples.

The results of missing values and irregular measures can be found in Figure 4.8 and Figure 4.9 respectively. As can be seen, IGAMT and ablation models contains time embedding have better performance than models without time embedding (i.e., baseline models and

$IGAMT\_w/o\_emb$). It is intuitive to explain for irregular measures, since time embedding can capture irregular characteristics and enhance the performance. For missing values, it reflects two points: 1) baseline architectures are not complicated enough to capture missing value characteristics, and 2) time embedding and the grasp of irregular characteristic can enhance model's capability of learning missing values.

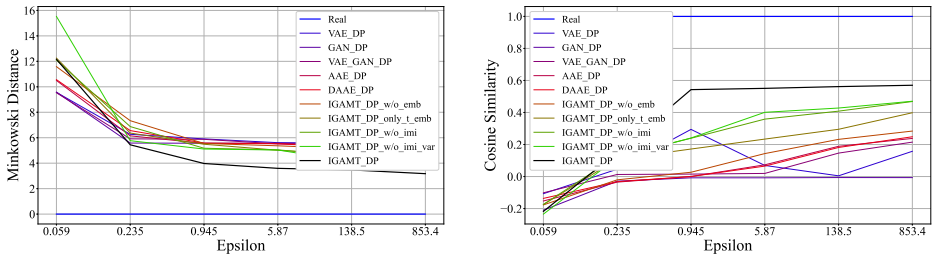


Figure 4.10: Overall Minkowski distance and cosine similarity on different models with different ϵ.

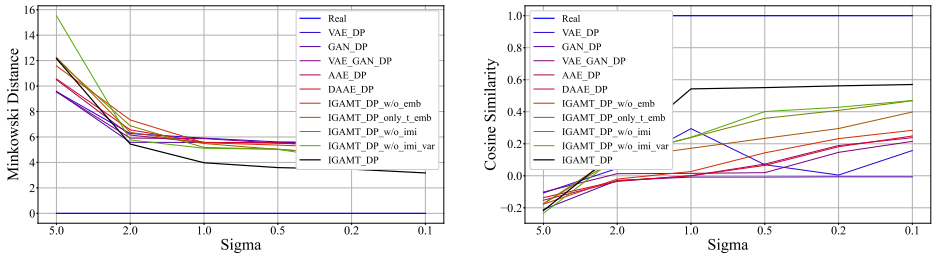


Figure 4.11: Overall Minkowski distance and cosine similarity on different models with different σ.

4.3.6 Differential Privacy

We plot the overall Minkowski distance and cosine similarity of models with different DP budget ϵ and different perturbation standard deviation σ. The results are shown in Figure 4.10 and Figure 4.11. These figures show that 1) when $\sigma < 2.0$ ($\epsilon > 0.235$), IGAMT outperforms all other models; 2) when $\sigma > 2.0$, the DP budget $\epsilon < 0.235$, which brings overwhelmed randomization to training process, and all models suffer, esp. those models having complex structures.

In summary, the experimental results w.r.t. overall Minkowski distance and cosine similarity show that IGAMT can achieve better performance than all other models.

Chapter 5

Generating Adversarial Examples with Distance Constrained Adversarial Imitation Networks

5.1 Overview

Recent studies have shown that neural networks are vulnerable to adversarial examples that are designed by adding small perturbations to clean examples in order to trick the classifier to misclassify. Various approaches based on optimization have been proposed for generating adversarial examples with minimal perturbation. Model training based methods such as Adversarial Transformation Network (ATN) provide a fundamentally new way to directly transform an input into an adversarial example, which promises fast generation of adversarial examples. However, the adversarial examples may have suboptimal quality with significantly large perturbations or low attack success rate at small perturbations. In this paper, we propose a distance constrained **A**dversarial **I**mitation **N**etwork (**AIN**), which enhances ATN and is capable of generating both targeted and untargeted examples with an explicit distance constraint. AIN can not only generate large scale adversarial examples ef-

ficiently as achieved in ATN, but also imitate the behavior of state-of-the-art optimization-based methods, hence achieving improved quality. Extensive experiments show that AIN significantly outperforms ATN and other Generative Adversarial Networks (GAN) based methods in the quality of generated adversarial examples, and is much more efficient than optimization based methods while achieving comparable quality.

5.2 Adversarial Imitation Networks

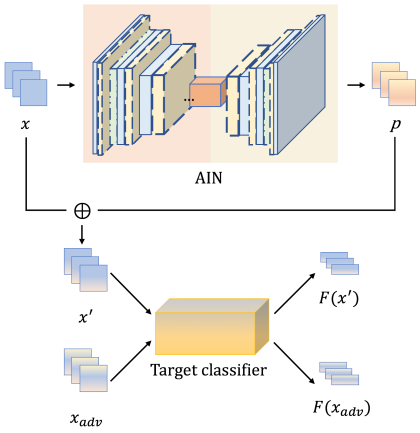


Figure 5.1: The training framework of AIN

In this section, we present our proposed Adversarial Imitation Networks (AIN). Fig. 5.1 shows the training framework of AIN. Given a clean example $\mathbf{x}$ and an adversarial example $\mathbf{x_{adv}}$ that can be generated by any of the state-of-the-art optimization based methods, AIN generates a perturbation $\mathbf{p}$ which can be used to create an adversarial example $\mathbf{x'} = \mathbf{x} + \mathbf{p}$. Both $\mathbf{x'}$ and $\mathbf{x_{adv}}$ are then fed into the target classifier $\mathbf{F}$ to get their classification logits $\mathbf{F}(\mathbf{x'})$ and $\mathbf{F}(\mathbf{x_{adv}})$ respectively. $\mathbf{F}$ is a well-trained white box model, whose parameters are fixed during the training. AIN is trained in such a way that 1) the distance between $\mathbf{x}$ and $\mathbf{x'}$ is bounded and minimized (distance constraint); 2) $\mathbf{x'}$ imitates the example $\mathbf{x_{adv}}$ both in its feature space and its classification output (imitation); and 3) $\mathbf{x'}$ will be misclassified by the target classifier (misclassification). Next, we present the objective function and the architecture of AIN in details followed by the training framework.

5.2.1 Objective Function

The key innovation of AIN is an imitation based objective function to have the output perturbation imitate the behavior of adversarial examples $\mathbf{x_{adv}}$ provided by optimization based attacks. Given the goals of the adversarial example attacks, our objective function $\mathbf{L}$ consists of the distance loss $\mathbf{L_d}$, the imitation loss $\mathbf{L_m}$, and the misclassfication loss $\mathbf{L_c}$:

$$\mathbf{L} = \beta_d \mathbf{L_d} + \beta_m \mathbf{L_m} + \beta_c \mathbf{L_c}, \tag{5.1}$$

where β_d, β_m and β_c denote positive coefficients that control the contributions of each loss.

Distance loss. AIN generates perturbations $\mathbf{p}$ and crafts $\mathbf{x'} = \mathbf{x} + \mathbf{p}$. $\mathbf{L_d}$ minimizes the L_2 distance between $\mathbf{x}$ and $\mathbf{x'}$, i.e.

$$\mathbf{L_d} = \|\mathbf{p}\|_2. \tag{5.2}$$

We impose an explicit element-wise bound $\mathbf{B}$ on $\mathbf{p}$ by clipping range $[-\mathbf{B}, \mathbf{B}]$, where each element of $\mathbf{B}$ is a small positive value which bounds $\|\mathbf{p}\|_\infty$. This explicit bound ensures a perturbation distance guarantee, and also makes it directly comparable with optimization based L_∞ attacks such as FGSM, i-FGSM, and C&W:

$$\mathbf{p} = \min(\max(\mathbf{z}, -\mathbf{B}), \mathbf{B}), \tag{5.3}$$

where $\mathbf{z}$ is the output of the last layer.

Imitation loss. The adversarial examples $\mathbf{x_{adv}}$ generated from optimization based attacks are provided during the model training. $\mathbf{L_m}$ forces $\mathbf{x'}$ to imitate the example $\mathbf{x_{adv}}$ both in its feature space and its classification output,

$$\mathbf{L_m} = \beta_{m,1} \|\mathbf{x'} - \mathbf{x_{adv}}\|_2 + \beta_{m,2} \|\mathbf{F}(\mathbf{x'}) - \mathbf{F}(\mathbf{x_{adv}})\|_2, \tag{5.4}$$

where $\mathbf{F}$ is the target classifier, $\mathbf{F}(\cdot)$ represents the classification output, and $\beta_{m,1}$ and $\beta_{m,2}$ denote positive coefficients that control the contributions of the two parts.

Misclassification loss. For targeted attacks, our goal is to classify $\mathbf{x'}$ as the target class t, i.e. maximize $\mathbf{F}(\mathbf{x'})_t$, where $\mathbf{F}(\cdot)_t$ represents the logits value on the position t. To further

reinforce this objective, and facilitate the optimization, we incorporate an additional objective to minimize the output probability for all classes other than t, i.e. minimize $\mathbf{F}(\mathbf{x}')_s$, where $s = \text{argmax}_{i \in \{0,\ldots,n-1\}-\{t\}} \mathbf{F}(\mathbf{x}')_i$. Therefore, we can formulate $\mathbf{L_c}$ for targeted attacks as:

$$\mathbf{L_c}^{targeted} = \mathbf{F}(\mathbf{x}')_s - \mathbf{F}(\mathbf{x}')_t. \tag{5.5}$$

For untargeted attacks, our goal is to have the classifier misclassify the adversarial example, i.e. to minimize $\mathbf{F}(\mathbf{x}')_y$, where y is the original class label of the clean example. Similar to targeted attacks, to facilitate this optimization, we incorporate an additional objective to maximize the output probability for all classes other than y, i.e. maximize $\mathbf{F}(\mathbf{x}')_s$, where $s = \text{argmax}_{i \in \{0,\ldots,n-1\}-\{y\}} \mathbf{F}(\mathbf{x}')_i$. Therefore, we can formulate $\mathbf{L_c}$ for untargeted attacks as:

$$\mathbf{L_c}^{untargeted} = \mathbf{F}(\mathbf{x}')_y - \mathbf{F}(\mathbf{x}')_s. \tag{5.6}$$

5.2.2 Architecture

AIN is based on a conditional self-attention autoencoder. The key idea is to utilize the categorical information of the original class label y together with the input $\mathbf{x}$ to improve the performance of generated adversarial example. In this section, we present our label conditioning approach followed by the structure of AIN and random noise injection to allow more diverse adversarial example generation.

Label conditioning. Traditionally, concatenation based approach incorporates label information into generative model by directly concatenating one-hot encoding of y to inputs. However, since the dimensionality and domain of the one-hot encoded $\mathbf{y}$ can be significantly different from the feature space of $\mathbf{x}$, direct concatenation can destroy the space dependencies among each pixel and weaken the functionality of the convolutional layers in AIN. To solve this challenge, we propose a novel method that can incorporate conditional information into AIN while preserving space dependencies.

We build a *conditional state extractor* to transform the one-hot encoded $\mathbf{y}$ into the

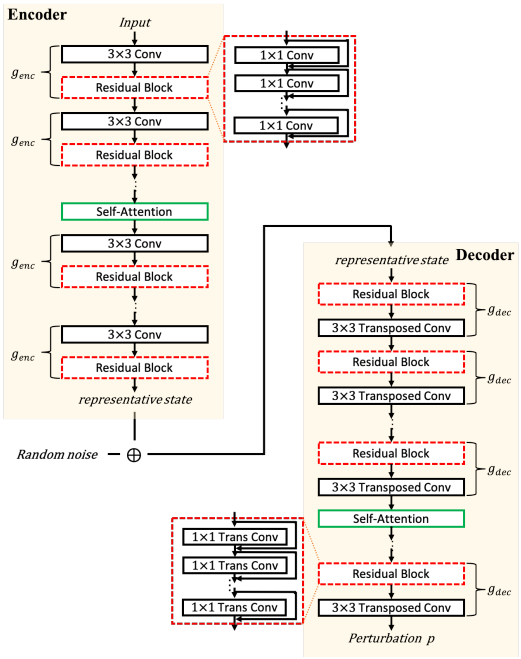


Figure 5.2: The architecture of AIN

conditional state $\mathbf{h_c}$ and set $\mathbf{h_c} \in \mathbb{R}^{h \times w}$, where h and w are the height and width of input images $\mathbf{x}$. We use a basic autoencoder as the conditional state extractor, which takes input $\mathbf{y}$ and generates output $\hat{\mathbf{y}}$ and is trained by optimizing the cross-entropy loss between $\mathbf{y}$ and $\hat{\mathbf{y}}$. We also leverage the $sigmoid$ activation function and the layer normalization in the extractor to constrain the numerical value of $\mathbf{h_c}$, so that it has the same domain as $\mathbf{x}$. $\mathbf{h_c}$ is then concatenated to $\mathbf{x}$ along the channel dimension to produce the concatenated input $\mathbf{x_c}$ while still preserving the space dependencies of $\mathbf{x}$.

Block structure. AIN consists of two parts, a convolutional encoder and a convolutional decoder. As shown in the left of Fig. 5.2, for the encoder, each convolutional layer (black rectangle) is followed by a residual block (red rectangle). The residual technique [40] has been proved to avoid the gradient vanishing problem that usually occurs in deep networks. As shown in the top middle of Fig. 5.2, the residual block is formed by several convolutional layers along with residual operations and can guarantee that the shapes of its input and output remain the same. A self-attention block (green rectangle) is added before one convolutional-residual structure to capture the correlation among pixels of the input. The

structure of the self-attention block we used here is from [97]. It facilitates the encoder to efficiently encode the inputs into the representative states, by effectively capturing the long-range dependency and the non-local correlation among pixels.

The decoder has a similar architecture, consisting of multiple residual and transposed convolutional structures, as shown in the right of Fig. 5.2. We also incorporate a self-attention block to efficiently decode the representative states into the reconstructed perturbations.

Random noise injection. Both ATN and AdvGAN can only generate one perturbation for one clean example in the testing phase. As a result, their capability of generating large scale adversarial examples is limited by the availability of clean examples. AIN extends this capability by injecting random noise into the central representative state, which is shown in Fig. 5.2. Because the random noise is injected in both training and testing phases, the adversarial examples generated by AIN differ from each other in different runs but all satisfy the distance constraint and the goal of misclassification. Therefore, AIN can generate even larger scale adversarial examples compared to ATN and AdvGAN, esp. when clean examples are limited.

5.2.3 Training Framework

Recall that $\mathbf{L_d}$ and $\mathbf{L_c}$ of our objective function optimize the distance given a bound and misclassification respectively. Since the two goals contradict with each other, directly optimizing the objective function is difficult and unstable. Therefore, we propose a progressive and adaptive training framework to train AIN in a stable and effective way, by starting from a loose distance bound, and progressively adapting to a tight distance bound.

Bound Decay

We first illustrate the motivation of the proposed progressive training in Fig. 5.3. The red and blue points represent images of two different classes, the dashed black line denotes the

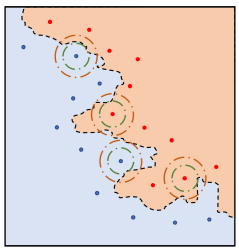


Figure 5.3: Illustration of the impact of distance bound $\mathbf{B}$ on optimization

classification boundary, the green circle denotes the adversarial example space for a tight distance bound $\mathbf{B_1}$ and the brown circle for a loose bound $\mathbf{B_2}$. We can see that it is difficult to find an adversarial example given the tight bound (green circle) that is misclassified, while it is much easier given a loose bound (brown circle). When an adversarial example is found with loose bound, we can gradually tighten the bound which will help us eventually obtain an adversarial example within the green circle.

We use three different decay methods adopted from learning rate decay [96] to decay the distance bound:

i) Step decay decreases the bound by a decay rate d every few epochs:

$$\mathbf{B}_i = \mathbf{B}_0 \times d^{\lfloor i/v \rfloor} \tag{5.7}$$

where i is the epoch index, v is the step size, $\mathbf{B}_0$ is the initial bound, $\mathbf{B}_i$ is the bound in the i-th epoch, and d ($0 < d < 1$) is the decay rate.

ii) Exponential decay decreases the bound by an exponential function every epoch:

$$\mathbf{B}_i = \mathbf{B}_0 \times e^{d \times i} \tag{5.8}$$

where d ($d < 0$) is the decay rate.

iii) Time-based decay decreases the bound linearly in the initial iterations (when $i < 1/d$) and then with a dampened decay rate in the later iterations:

$$\mathbf{B}_i = \mathbf{B}_0/(1 + d \times i) \tag{5.9}$$

where d ($d > 0$) is the decay rate.

Adaptive Bound Decay

As shown in Fig. 5.3, given a tighter distance bound, it is more difficult to find an adversarial example for misclassification. However, during our experiment studies, we observed that the misclassification or attack success rate is not monotonically or proportionally decreasing as the bound decays. This can be contributed to the fact that the bound decay may dramatically change the possible adversarial example space in certain ranges, but less in others. A decay rate that dramatically changes the adversarial example space can make it difficult for AIN to find optimal perturbations. On the other hand, a decay rate that does not sufficiently change the search space will make the training slow to converge. To address this, we propose an adaptive bound decay, which adaptively adjusts the bound decay rate d according to the relative change of the attack success rate r.

Specifically, the relative change between the current epoch's attack success rate S_1 and the previous epoch's S_0 is defined as r, i.e., $r = \frac{S_0 - S_1}{S_1}$. r is compared with two hyper-parameters T_{up} and T_{low} ($T_{up} > T_{low}$), both of which have a range from 0 to 1 and are close to 0. When r is larger than T_{up}, it means that the relative change of the attack success rates caused by the bound decay is too large. Therefore, we roll back our model to the previous epoch and increase the decay rate as $d = (1 + \gamma_1)d$ to slow down the decay, where γ_1 is a positive hyper-parameter close to 0. On the other hand, when r is smaller than T_{low}, it means that relative change of the attack success rates is too small. We can decrease the decay rate as $d = (1 - \gamma_2)d$ to speedup the decay, where γ_2 is also a positive hyper-parameter close 0. After every epoch's training, either the model rolls back to the previous one or the bound B is updated by one of the decay methods with the decay rate d. The details of the adaptive training process is presented in Algorithm 2.

Bound Constraint at Inference. Bound constraint $\mathbf{B}$ works as a hyper-parameter and progressively decreases during training. Once training phase ends, a well-trained AIN and a final bound constraint $\mathbf{B}_{final}$ are obtained. Therefore, at inference phase, AIN can be applied with any distance constraint $\mathbf{B}_{infer}$ to generate adversarial examples as long as

Algorithm 2: AIN training with adaptive bound decay

Input: initial bound B, total training epoch N, thresholds T_{up}, T_{low}, bound decay function f_{decay}, bound decay rate d, rate changing coefficients γ_1, γ_2, training dataset D_{train}, validation dataset D_{valid}

```
i = 0;
S_0 = 1.0 ;
while i < N do
    mini-batch training AIN on D_train with perturbation bound B;
    calculate validation attack success rate S_1;
    r = (S_0 - S_1) / S_1;
    if r > T_up then
        roll back AIN to previous epoch's setting;
        d = (1 + γ_1) d;
    else
        if r < T_low then
            d = (1 - γ_2) d;
        end
        B = f_decay(B, d);
        S_0 = S_1;
    end
end
```

$\mathbf{B}_{infer} \geq \mathbf{B}_{final}$.

5.3 Experiments

In this section, we present our experimental studies evaluating AIN in comparison with existing methods using the benchmark MNIST [54] and tiny-imagenet datasets [22].

Implementation details. All models are implemented using Tensorflow 1.14 [1] and trained with a system equipped with Nvidia GTX 1080 Ti GPU. We use Adam [49] for optimization. AIN uses adversarial images generated by i-FGSM for imitation. The initial learning rate is set to 1×10^{-3} on MNIST and 5×10^{-4} on tiny-imagenet with a 0.99 decay rate. By default, we use exponential decay in our training framework, while we compare different decay methods in one of the experiments. The initial distance bound B is set to 0.8. The decay rate d is set to -0.02, γ_1 and γ_2 are set to 0.5. The threshold $T_{up} = 1 \times 10^{-4}$ and $T_{low} = 1 \times 10^{-5}$. We use the open source code of AdvGAN, i-FGSM and C&W to

conduct experiments. In this paper, we apply L_∞ attacks of i-FGSM and C&W to target models. We implemented ATN ourselves since the source code for ATN is not available. The ATN results reported in our experiments are the best results we achieved. The code of this paper is available at `https://github.com/Emory-AIMS/AIN`.

Comparison. For targeted attacks, we compare AIN with model based methods including ATN and AdvGAN and optimization based attacks. For untargeted attacks, we compare AIN with AdvGAN and optimization based methods since the current implementation of ATN does not support untargeted attacks. For optimization based methods, we use i-FGSM and C&W, which are the most popular and state-of-the-art attack methods. We note that C&W is used as a reference for the optimal result one can achieve but not directly comparable with AIN since it incurs a significant computation overhead. We perform ablation studies and evaluate the benefit of 1) different components of our objective function including distance loss, imitation loss, and misclassification loss by comparing AIN with AIN trained without distance loss, AIN with only distance loss, AIN trained without imitation, AIN with only imitation, AIN trained without misclassification loss, and AIN with only misclassification loss; 2) different components of AIN architecture including self-attention, residual technique, and label conditioning by comparing AIN with AIN without self-attention, AIN without residual, and AIN without label conditioning; 3) different adversarial examples to be imitated in AIN training by comparing AIN imitating i-FGSM, and AIN imitating i-FGSM and C&W, and 4) our adaptive training framework. To further evaluate the utility and diversity of the adversarial examples generated by AIN besides the quality, we also conduct adversarial training experiments to compare performance of adversarial training using examples generated by AIN and i-FGSM respectively.

Evaluation metrics. We evaluate the quality of adversarial examples in terms of perturbation distance and attack success rate. We evaluate the efficiency of the methods by the average time cost for generating each adversarial example. We also report the time for training AIN. For the utility of the adversarial examples for adversarial training, we evalu-

ate models obtained from adversarial training in terms of classification accuracy on clean examples and different adversarial examples.

Target models. In our paper, AINs are trained with white-box access to the target classifier. For MNIST dataset, the target classifier is a 24-layer deep convolutional network trained by ourselves, which achieves 98.9% classification accuracy on the validation dataset. For tiny-imagenet dataset, the target classifier is a pre-trained Resnet-18 [40] network which achieves 80.5% validation accuracy.

5.3.1 Targeted Attacks

We conduct targeted attacks on MNIST and tiny-imagenet datasets to compare the performace of AIN with ATN, AdvGAN, i-FGSM and C&W. Our experiments show that AIN 1) significantly outperforms ATN in terms of perturbation scale and attack success rate, 2) slightly outperforms AdvGAN in terms of L_∞ and attack success rate, and 3) achieves comparable performance with i-FGSM and C&W. More importantly, AIN can generate adversarial examples with different distance bound (thanks to its explicit distance constraint) and allow flexible tradeoff between perturbation scale and attack success rate compared to AIN and AdvGAN.

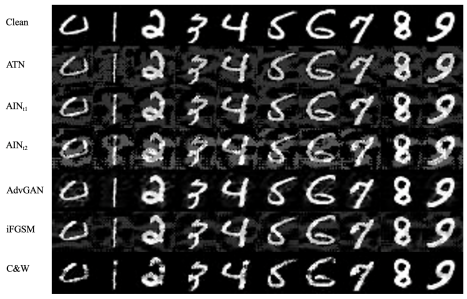


Figure 5.4: Targeted adversarial examples of different attacks on MNIST

MNIST experiments. For MNIST experiments, we set the targeted class as 8, which means that all attacks attempt to generate targeted adversarial images to be classified as 8. Fig. 5.4 shows sample adversarial examples on MNIST for the different models under selected settings. The first row is clean images and the second row is adversarial images

Table 5.1: Comparison of L_∞ distances, targeted attack success rate and average generation time on MNIST among different attacks

Attacks	L_∞	**Attack success rate**	**Avg time**
ATN	0.200	42.22%	0.003s
AIN_{t1}	0.200	90.94%	0.003s
AIN_{t2}	0.300	100.00%	0.003s
AdvGAN	0.300	99.40%	0.0008s
i-FGSM	0.200	90.10%	0.007s
C&W	0.180	98.00%	112s

generated by ATN. The third to the fourth rows represent images generated by AIN with the perturbation bound (L_∞ distance) of 0.2 and 0.3 respectively. The two AIN settings are aimed to show the trade-off between perturbation scale and attack success rate. The last three rows are AdvGAN, i-FGSM, and C&W respectively. AIN images are comparable to i-FGSM which is a successful imitation of i-FGSM.

Table 5.1 shows the average values of L_∞, attack success rate, and average generation time over the test dataset for the corresponding models illustrated in Fig. 5.4. Compared with ATN, with more strict perturbation bound (AIN_{t1}), AIN can achieve much higher attack success rate with same L_∞ distance. With looser bound (AIN_{t2}), AIN achieves 100.00% attack success rate. Compared with AdvGAN, AIN (AIN_{t2}) achieves better attack success rate with same L_∞ and similar average time cost. Compared with i-FGSM, AIN (AIN_{t1}) achieves higher attack success rate with the same L_∞ and less average time cost. C&W achieves a higher attack success rate with smaller L_∞. However, C&W costs too much time to generate one adversarial example, around 112 seconds, which can be prohibitively high for real time or large scale generation.

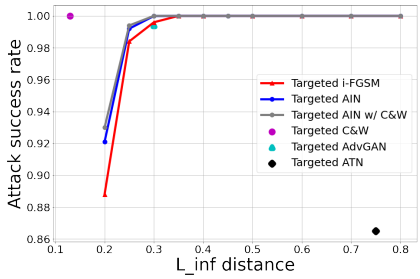


Figure 5.5: The targeted attack success rate vs. L_∞ distance among different attacks

Fig. 5.5 shows a more comprehensive comparison of the different models with respect to varying perturbation distances. In addition to AIN which uses all i-FGSM examples as input for imitation, we included a variant AIN w/ C&W which is trained with a small amount of C&W examples (less than 10%) together with i-FGSM examples (more than 90%)). Compared with AIN, AIN w/ C&W has higher attack success rate, which shows that the adversarial examples used for imitation can affect the capability of AIN and more "powerful" imitated adversarial examples such as those generated by C&W can improve the attack success rate of AIN. Compared with ATN, AdvGAN, and i-FGSM, AIN outperforms them on any L_∞ distances. Note that ATN and AdvGAN do not allow explicit tradeoff by different distance bounds.

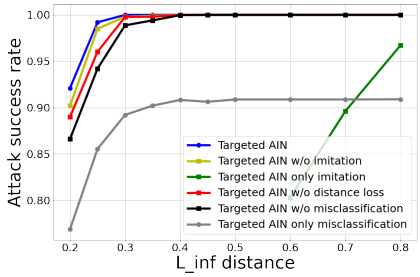


Figure 5.6: The targeted attack success rate vs. L_∞ distance among AINs with different loss

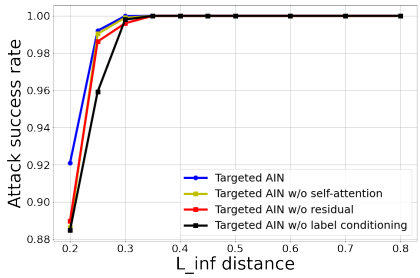


Figure 5.7: The targeted attack success rate vs. L_∞ distance among AINs with different architectures

Fig. 5.6 shows a comparison of AINs with different losses including: AIN w/o imitation, AIN with only imitation, AIN w/o distance loss, AIN w/o misclassification loss, and AIN with only misclassification loss. AIN with only distance loss has attack success rate lower than 0.1 at any L_∞ distances, which is not shown here. Comparing AIN with these ablation cases, AIN has higher attack success rate at any different L_∞ distance, which

proves the effectiveness of different parts of the loss function and the effectiveness of the combined loss.

Fig. 5.7 shows a comparison of AINs with different architectures including: AIN w/o self-attention, AIN w/o residual technique, and AIN w/o label conditioning. Comparing AIN with these ablation cases, AIN has higher attack success rate at any different L_∞ distance, which proves the effectiveness of these different technique applied to the architecture and the effectiveness of the combination.

The previous experiments use 8 as the targeted class. We also conduct experiments with different targeted labels. Figure 5.8 shows the results of targeted AIN attacks with randomly selected classes. We observe that AIN is a general attack model that works for all different targeted attacks.

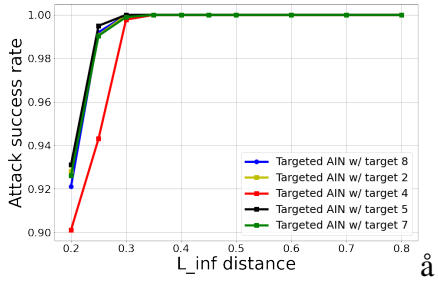


Figure 5.8: The targeted attack success rate vs. L_∞ distance among AINs with different targeted labels

Figure 5.9: Targeted adversarial examples of different attacks on tiny-imagenet

Tiny-imagenet experiments. For Tiny-imagenet experiments, we set the targeted class as "espresso". The sample generated adversarial examples are shown in Fig. 5.9.

If we look at the highlighted adversarial example generated by AIN in Fig. 5.10, we can see that there is a pattern of a cup of espresso in the left bottom corner. This pattern shows that AIN can learn to add specific noises on the clean image to incur the misclassification.

Figure 5.10: Detail of a targeted adversarial examples by AIN

Table 5.2: Comparison of L_∞ distances and targeted attack success rate on tiny-imagenet among different attacks

Attacks	L_∞	**Attack success rate**	**Avg time**
ATN	0.10	32.82%	0.02s
$\mathrm{AIN}_{\mathrm{t1}}$	0.10	97.66%	0.02s
$\mathrm{AIN}_{\mathrm{t2}}$	0.28	100.00%	0.02s
AdvGAN	0.28	97.61%	0.006s
i-FGSM	0.01	97.60%	0.08s
C&W	0.009	100.0%	558.8s

Table 5.2 shows the details of these adversarial examples. The second and the third rows represent images generated by AIN with the perturbation bound (L_∞ distance) of 0.1 and 0.28 respectively. As can be seen, compared with i-FGSM, both $\mathrm{AIN}_{\mathrm{t1}}$ and $\mathrm{AIN}_{\mathrm{t2}}$ have larger L_∞ distances, and better attack success rate. Compared with AdvGAN, $\mathrm{AIN}_{\mathrm{t2}}$ achieves better attack success rate with same L_∞. Compared with ATN, AIN has significantly higher attack success rate with same L_∞. AIN, like ATN, is 4 times faster than i-FGSM and 1000 orders of magnitude faster than C&W.

We also conduct experiments for targeted AIN with different targeted labels on tiny-imagenet. Table 5.3 shows the results of different targeted AIN attacks on tiny-imagenet with 3 more randomly selected classes among 200 in addition to “expresso”, where $\mathrm{AIN}_{\mathrm{t8}}$, $\mathrm{AIN}_{\mathrm{t71}}$, $\mathrm{AIN}_{\mathrm{t99}}$, and $\mathrm{AIN}_{\mathrm{t183}}$ target “espresso”, “teapot”, “butcher shop, meat market”, and ”tailed frog, etc”, respectively. The results show AIN is generalizable for different targets.

Table 5.3: Comparison of L_∞ distances and targeted attack success rate on tiny-imagenet among different attacks

Attacks	L_∞	**Attack success rate**	**Avg time**
$\mathrm{AIN}_{\mathrm{t8}}$	0.10	97.66%	0.02s
$\mathrm{AIN}_{\mathrm{t71}}$	0.10	98.02%	0.02s
$\mathrm{AIN}_{\mathrm{t99}}$	0.10	96.97%	0.02s
$\mathrm{AIN}_{\mathrm{t183}}$	0.10	97.74%	0.02s

5.3.2 Untargeted Attacks

As ATN is not able to generate untargeted adversarial examples, we compare the performance of AIN with AdvGAN, i-FGSM, and C&W. Our experiments show that AIN outperforms AdvGAN and has comparable performance with i-FGSM and C&W, especially on MNIST.

MNIST experiments. Fig. 5.11 compares the adversarial examples generated by AIN, AdvGAN, i-FGSM and C&W on MNIST dataset. The two AIN models shown have the perturbation bound of 0.2 and 0.3 respectively. As can be seen from Fig. 5.11 and Table 5.4, C&W outperforms others but costs 1000 orders of magnitude more time for generation. Compared with AdvGAN, AIN ($\mathrm{AIN_{u2}}$) can achieve 100% attack success rate with same L_∞. Compared with i-FGSM, AIN ($\mathrm{AIN_{u1}}$) also achieves higher attack success rate with same distance.

Figure 5.11: Untargeted adversarial examples of different attacks on MNIST

Table 5.4: Comparison of L_∞ distances and untargeted attack success rate on MNIST among different attacks

Attacks	L_∞	**Attack success rate**	**Avg time**
$\mathrm{AIN_{u1}}$	0.200	98.70%	0.003s
$\mathrm{AIN_{u2}}$	0.300	100.00%	0.003s
AdvGAN	0.300	99.6%	0.0008s
i-FGSM	0.200	97.05%	0.006s
C&W	0.13	100.00%	98.1s

Fig. 5.12 shows a more comprehensive comparison among AIN, AIN w/ C&W, AdvGAN, i-FGSM, and C&W in test attack success rate versus different L_∞ distances. The results verify that 1) AIN can achieve comparable performance with C&W, 2) AIN achieves

higher attack success rate than AdvGAN and i-FGSM with the same L_∞ distance, and 3) training AIN with imitation of more "powerful" adversarial examples can improve the attack success rate of AIN.

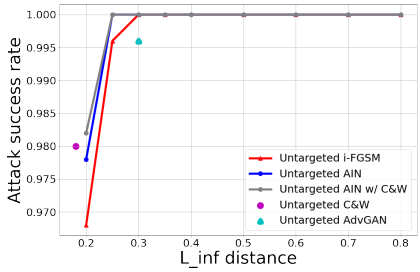


Figure 5.12: The untargeted attack success rate vs. L_∞ distance among different attacks

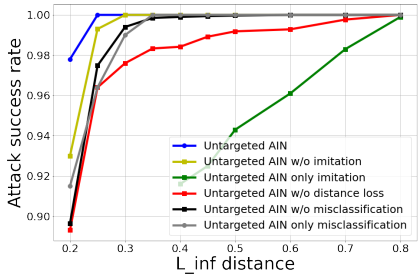


Figure 5.13: The untargeted attack success rate vs. L_∞ distance among AINs with different losses

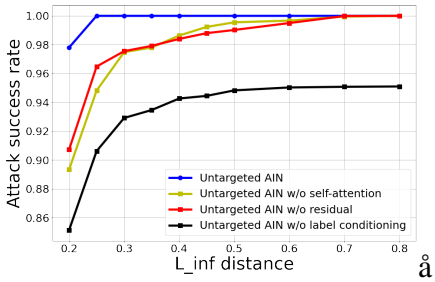


Figure 5.14: The untargeted attack success rate vs. L_∞ distance among AINs with different architectures

Fig. 5.13 shows a comparison of AINs with different losses including: AIN w/o imitation, AIN with only imitation, AIN w/o distance loss, AIN w/o misclassification loss, and AIN with only misclassification loss. AIN with only distance loss has attack success rate lower than 0.08 at any L_∞ distances, which is not shown here. Comparing AIN with these ablation cases, AIN has higher attack success rate at any different L_∞ distance, which again proves the effectiveness of different parts of the loss function and the effectiveness of the combined loss.

Fig. 5.14 shows a comparison of AINs with different architectures including: AIN w/o self-attention, AIN w/o residual technique, and AIN w/o label conditioning. Comparing AIN with these ablation cases, AIN has higher attack success rate at any different L_∞ distance, which proves the effectiveness of these different technique applied to the architecture and the effectiveness of the combination.

Tiny-imagenet experiments. We conduct similar experiments on tiny-imagenet to compare the performance of AIN, AdvGAN, i-FGSM, and C&W. The generated adversarial examples are shown in Fig. 5.15. Table 5.5 shows the details of these adversarial examples. Compared with AdvGAN, AIN achieves slightly higher attack success rate with same L_∞. Compared with i-FGSM and C&W, AIN has larger L_∞ distances with comparable attack success rate, but much faster generation time.

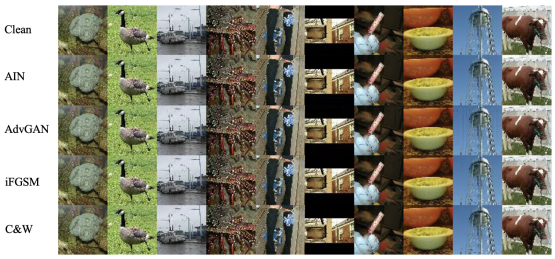


Figure 5.15: Untargeted adversarial examples of different attacks on tiny-imagenet

Table 5.5: Comparison of L_∞ distances and untargeted attack success rate on tiny-imagenet among different attacks

Attacks	L_∞	**Attack success rate**	**Avg time**
AIN	0.10	99.92%	0.02s
AdvGAN	0.10	99.60%	0.006s
i-FGSM	0.01	100.00%	0.064s
C&W	0.004	100.00%	486.2s

5.3.3 Training Framework

In this section, we conduct experiments to evaluate the effectiveness of the progressive training framework using MNIST dataset. We show that 1) the bound decay can significantly increase the attack success rate of the adversarial examples given the same bound,

and 2) the adaptive bound decay further improves the performance.

Bound decay. To evaluate the effectiveness of the bound decay, we train several AINs starting from different initial loose perturbation bounds separately and decay the bounds to final tight bound 0.26. We also directly train an AIN with an initial and fixed bound 0.26 as the non-progressive benchmark. Fig. 5.16 shows the test attack success rate with respect to varying initial bounds for progressive and non-progressive training respectively.

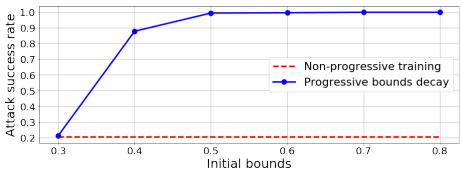


Figure 5.16: The attack success rate for non-progressive training and progressive training with different initial bounds

We observe that 1) AIN under progressive training achieves significantly higher attack success rate, 2) AIN with initial bound larger or equal to 0.5 can outperform non-progressive training by almost 80% higher attack success rate, and 3) an initial bound larger than 0.5 helps the test attack success rate marginally but would increase the training time.

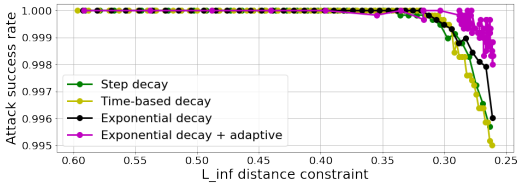


Figure 5.17: The attack success rate as L_∞ distance constraint decreases during the model training for different frameworks

Adaptive bound decay. We compare the the different training processes of bound decay with and without adaptive technique in Fig. 5.17. The x-axis represents the L_∞ distance between adversarial examples and clean examples. It is reversed to show the decrease of L_∞ distance as we move along training epochs. The y-axis represents the attack success rate on the validation set during the training. This figure shows that 1) for progressive training, different bound decay approaches can affect the attack success rate and exponential decay

has the best performance overall, 2) the adaptive technique changes the perturbation bound back and forth when the distance becomes smaller, and 3) the exponential decay approach with adaptive technique outperforms non-adaptive approaches.

5.3.4 Adversarial Training

In this section, we conduct experiments to further evaluate the utility of adversarial examples for adversarial training. We train AIN by a target model, then use AIN to generate adversarial examples to be used for adversarial training. We show that 1) compared with adversarial training with i-FGSM, adversarial training with AIN can achieve comparable or better model accuracy on tiny-imagenet and better accuracy on MNIST, and 2) random noise injection in AIN's central hidden layer can benefit adversarial training.

Table 5.6: Comparison of test classification accuracy on MNIST among models from different adversarial training

Models	**Clean acc**	**AIN acc**	**i-FGSM acc**
AIN	98.63%	98.68%	96.48%
$\mathrm{AIN_{nr}}$	98.20%	98.42%	96.06%
i-FGSM	98.62%	94.84%	96.47%

Table 5.7: Comparison of test classification accuracy on tiny-imagenet among models from different adversarial training

Models	**Clean acc**	**AIN acc**	**i-FGSM acc**
AIN	77.64%	75.56%	75.86%
$\mathrm{AIN_{nr}}$	76.02%	75.13%	73.46%
i-FGSM	78.24%	71.48%	76.24%

Table 5.6 and Table 5.7 show the results on MNIST and tiny-imagenet respectively. In both tables, AIN denotes adversarial training with adversarial examples form AIN, $\mathrm{AIN_{nr}}$ denotes adversarial training with adversarial exampels from AIN without noise injection in central hidden layer, and i-FGSM denotes adversarial training with adversarial examples form i-FGSM. Clean acc, AIN acc, i-FGSM acc denote the classification accuracy on clean examples, AIN adversarial examples, i-FGSM adversarial examples respectively.

As can be seen from these tables, compared with adversarial training with i-FGSM, training with AIN achieves 1) comparable or better test classification accuracy on clean examples, 2) higher classification accuracy on adversarial examples generated by AIN, and 3) similar accuracy on adversarial examples generated by i-FGSM. Compared with $\mathrm{AIN}_{\mathrm{nr}}$, training with AIN achieves better performance, verifying the benefit of random noise injection in AIN.

5.3.5 AIN Training Time

We reported in Section 4.1 and 4.2 the average time for generating one adversarial example by AIN once AIN is trained, which is very fast. One potential concern about AIN is its training cost. We evaluate the itemized time cost for training AIN and using AIN for generating adversarial examples in the following three steps: step 1) generating adversarial examples by i-FGSM or C&W, step 2) using the i-FGSM or C&W adversarial examples to train AIN, and step 3) using AIN to generate adversarial examples.

Table 5.8: Comparison of time costs on MNIST among models with different adversarial examples

Models	**step 1**	**step 2**	**step 3**	**Total time**
AIN w/ C&W	447,269s	7,166s	162s	454,597s
C&W	6,054,913s	0s	0s	6,054,913s

Table 5.8 shows the itemized and total time cost of AIN w/ C&W in comparison with C&W on MNIST dataset. Note that traditional attack methods such as C&W and i-FGSM generate adversarial examples directly and do not require training, so we show the time under step 1 and total time. AIN w/ C&W uses a small volume $(4,000)$ of adversarial examples from C&W and a large volume $(50,000)$ of adversarial examples from i-FGSM for model training. AIN w/ only i-FGSM examples will cost almost negligible time on step 1. As can be seen, although AIN w/ C&W costs **7,166**s on step 2 and **162**s on step 3, the total time of AIN (**454,597**s) is significantly lower than what's required to generate sufficient number of adversarial examples using optimization methods directly (**6,054,913**s).

Note that AIN is trained by the targeted model. If the targeted model changes completely, for instance, structures of all layers are modified, we need to re-train AIN from scratch. The time cost for retraining AIN and generating new adversarial examples is still significantly lower than regenerating adversarial examples using optimization based attack algorithms like C&W as shown in Table 5.8. If the targeted network changes slightly, for instance, only one or a few layers are modified, we can take the previous AIN as a pre-trained model, use its parameters as the initialization of current AIN, and train an AIN for the new targeted network like "fine-tuning". Since the two targeted networks are similar, this "fine-tuning" will cost much less time than training from scratch.

Chapter 6

Achieving both Differential Privacy and Certified Robustness for Pre-trained Classifiers via Input Perturbation

6.1 Overview

Recent studies have shown that pre-trained classifiers are increasingly powerful to improve the performance on different tasks, e.g, neural language processing, image classification. However, adversarial examples from attackers can trick pre-trained classifier to misclassify. To solve this challenge, a reconstruction network is built before the public pre-trained classifiers to offer certified robustness and defend against adversarial examples through input perturbation. On the other hand, the reconstruction network requires training on the dataset, which incurs privacy leakage of training data through inference attacks. To prevent this leakage, differential privacy (DP) is applied to offer a provable privacy guarantee on training data through gradient perturbation. Most existing works employ certified robustness and DP independently and fail to exploit the fact that input perturbation designed to achieve certified robustness can achieve (partial) DP. In this paper, we propose perturba-

tion transformation to show how the input perturbation designed for certified robustness can be transformed into gradient perturbation during training. We propose Multivariate Gaussian mechanism to analyze the privacy guarantee of this transformed gradient perturbation and precisely quantify the level of DP achieved by input perturbation. To satisfy the overall DP requirement, we add additional gradient perturbation during training and propose Mixed Multivariate Gaussian Analysis to analyze the privacy guarantee provided by the transformed gradient perturbation and additional gradient perturbation. Moreover, we prove that Mixed Multivariate Gaussian Analysis can work with moments accountant to provide a tight DP estimation. Extensive experiments on benchmark datasets show that our framework significantly outperforms state-of-the-art methods and achieves better accuracy and robustness under the same privacy guarantee.

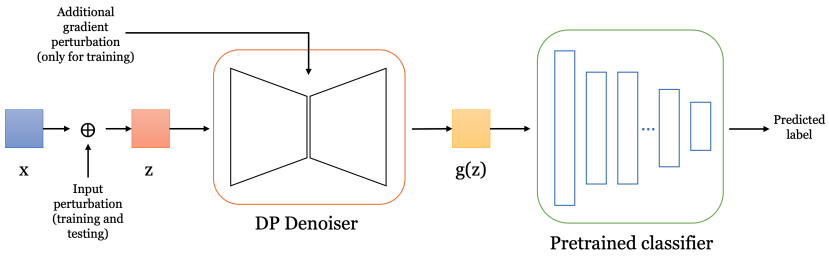


Figure 6.1: Framework of TransDenoiser: given a clean image $\mathbf{x}$, input perturbation is added to generate perturbed image $\mathbf{z}$. $\mathbf{z}$ is then reconstructed by denoiser g to generate $g(\mathbf{z})$, which is fed into the pre-trained classifier h for classification. The input perturbation on $\mathbf{x}$ is utilized to achieve certified robustness during testing. The denoiser is trained under DP by leveraging the input perturbation added on $\mathbf{x}$ and additional gradient perturbation during training.

6.2 TransDenoiser

In this section, we will present our proposed framework TransDenoiser, which can be applied before any pre-trained classifier to guarantee certified robustness and DP without retraining the pre-trained model. A denoiser is trained to guarantee certified robustness of the final model via randomized smoothing or input perturbation on the input, where the training process introduces the input perturbation for better accuracy of the randomized smoothed model. We transform the input perturbation into equivalent gradient perturbation

so that it can be quantified and composed with the explicit gradient perturbation for the DP guarantee.

6.2.1 Denoiser and Certified Robustness

As can be seen in Figure6.1, TransDenoiser is a denoising AE trained on input data with Gaussian perturbation. Similar to [74], this input Gaussian perturbation is used as randomized smoothing for certified robustness. Naively applying randomized smoothing on a pre-trained classifier without the denoiser gives very loose certification bounds because the pre-trained classifier is not trained to be robust to Gaussian perturbations of their input. The denoiser serves to "remove" this Gaussian perturbation and effectively reconstruct the input data like a pre-processing step before feeding input data into the pre-trained classifier while maintaining the benefit of certified robustness. The detailed proof of randomized smoothing can be found in [21], and we provide a brief proof in Appendix .3.

Different from [74], given input data $\mathbf{x}_{(i)}$ and perturbed data $\mathbf{z}_{(i)} = \mathbf{x}_{(i)} + \mathbf{b}_{(i)}$ with $\mathbf{b}_{(i)} \sim \mathcal{N}(0, \sigma^2 I)$, the objective function we use to optimize the denoiser contains the standard reconstruction MSE:

$$l(\mathbf{z}_{(i)}, \theta) = \|g(\mathbf{z}_{(i)}) - \mathbf{x}_{(i)}\|_2^2, \tag{6.1}$$

where l is the loss function, g denotes the denoiser, and θ is the parameter of the denoiser. The stability objective of [74] is not included in our objective function, because it requires both $\mathbf{x}_{(i)}$ and $g(\mathbf{z}_{(i)})$ to pass the pre-trained classifier h, which both incurs additional privacy cost and additional computation overhead when we calculate the transformation matrix. In this work, we assume $l(\mathbf{z}_{(i)}, \theta)$ is C-Lipschitz continuous, which is a mild and common assumption in existing works [11, 30].

6.2.2 Perturbation Transformation and Multivariate Gaussian Mechanism

In this section, we will introduce perturbation transformation and Multivariate Gaussian Mechanism (**MGM**) to analyze the DP guarantees of the input perturbation.

Perturbation transformation. As we introduced in 6.2.1, input Gaussian perturbation is utilized to achieve certified robustness for TransDenoiser. Although theoretically this perturbation is only required at the testing phase, almost all existing approaches in practice demand the randomization during training to improve the performance. Our strategy is to transform input perturbation into gradient perturbation during training and analyze the DP guarantee that input perturbation can offer. The crucial step of this transformation is the Taylor expansion of MSE loss $l(\mathbf{z}_{(i)}, \theta)$ at the data point $\mathbf{x}_{(i)}$, which is formulated as follows,

$$l(\mathbf{z}_{(i)}, \theta) = l(\mathbf{x}_{(i)}, \theta) + (\mathbf{z}_{(i)} - \mathbf{x}_{(i)})^{\intercal} \nabla_{\mathbf{x}_{(i)}} l(\mathbf{x}_{(i)}, \theta) + o(\mathbf{z}_{(i)} - \mathbf{x}_{(i)}) \tag{6.2}$$

Since the only mild constraint on $l(\mathbf{z}_{(i)}, \theta)$ is C-Lipschitz continuous, it is possible for the higher order terms $o(\mathbf{z}_{(i)} - \mathbf{x}_{(i)})$ to be negative. We denote the examples with non-negative higher order terms as "non-negative cases", and the others as "negative cases". In the rest of this section, we use the superscript "non" and "neg" to denote samples of "non-negative cases" and "negative cases", respectively. For "non-negative cases", we have the following lemma:

Lemma 1. *Given perturbed example* $\mathbf{z}_{(i)}^{non} = \mathbf{x}_{(i)}^{non} + \mathbf{b}_{(i)}$ *with* $\mathbf{b}_{(i)}^{(k)} \sim \mathcal{N}(0, \sigma^2)$*, and MSE loss* $l(\mathbf{z}_{(i)}^{non}, \theta)$ *is* C*-Lipschitz continuous. The gradient* $\nabla_\theta l(\mathbf{z}_{(i)}^{non}, \theta)$ *can be reformulated as the gradient with respect to the original sample with a gradient perturbation:*

$$\nabla_\theta l(\mathbf{z}_{(i)}^{non}, \theta) \geq \nabla_\theta l(\mathbf{x}_{(i)}^{non}, \theta) + \mathbf{p}_{(i)}, \tag{6.3}$$

where $\mathbf{p}_{(i)}$ *is the transformed perturbation with* $\mathbf{p}_{(i)} \sim \mathcal{N}(0, \mathbf{\Sigma}_{(i)})$, $\mathbf{\Sigma}_{(i)} = \mathbf{M}_{(i)}\sigma^2$, $\mathbf{M}_{(i)} = \mathbf{A}_{(i)}\mathbf{A}_{(i)}^{\intercal}$ *and* $\mathbf{A}_{(i)} = \mathbf{J}_\theta \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta)$.

The detailed proof of Lemma 1 can be found in Appendix .6. With Lemma 1, we find that the right-hand side is the lower bound of left-hand side, which means DP guarantee provided by transformed gradient perturbation $\mathbf{p}_{(i)}$ is the lower bound of that provided by input perturbation $\mathbf{b}_{(i)}$.

Multivariate Gaussian Mechanism. Since the transformed gradient perturbation $\mathbf{p}_{(i)}$ in equation (3) follows a multivariate Gaussian distribution with correlated elements, which is in contrast to the independent and isotropic Gaussian noise used in standard Gaussian mechanism for DP, we introduce Multivariate Gaussian Mechanism (**MGM**) to analyze DP of this multivariate Gaussian perturbation.

Theorem 3. *Multivariate Gaussian Mechanism. Let* $\mathcal{G} : \mathbb{R}^v \to \mathbb{R}^w$ *be an arbitrary* w*-dimensional function, and* $\Delta_{\mathcal{G}} = \max_{\mathcal{D},\mathcal{D}'} \|\mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')\|_2$. *A Multivariate Gaussian Mechanism* $\mathcal{M}$ *with the covariance* $\mathbf{\Sigma} \in \mathbb{R}^{w \times w}$ *adds noise to each of the* w *elements of the output. The mechanism* $\mathcal{M}$ *is* (ϵ, δ)*-DP, with*

$$\epsilon \in (0, 1],\ S_{min}(\mathbf{M})^{\frac{1}{2}}\sigma \geq \sqrt{2\ln(1.25/\delta)}\Delta_{\mathcal{G}}/\epsilon.$$

where $S_{min}(\mathbf{M})$ *is the minimum singular value of* $\mathbf{M}$ *and* $\mathbf{\Sigma} \triangleq \mathbf{M}\sigma^2$.

The proof of this theorem is in Appendix .5. With Theorem 3, multivariate Gaussian perturbation can be leveraged to preserve (ϵ, δ)-DP, and we have the following Corollary:

Corollary 1. *Given a multivariate Gaussian perturbation* $\mathbf{p} \sim \mathcal{N}(0, \mathbf{\Sigma})$, $\mathbf{\Sigma} = \mathbf{M}\sigma^2$, *the DP guarantee of the* ***MGM*** *is equivalent to that of a* ***GM*** *with its perturbation following a Gaussian distribution* $\mathcal{N}(0, S_{min}(\mathbf{M})\sigma^2)$, *where* $S_{min}(\mathbf{M})$ *is the minimum singular value of* $\mathbf{M}$.

Because the DP guarantee of **MGM** is equivalent to the **GM** by applying a transformed perturbation, the traditional DP analysis technique, e.g., moments accountant [2], can be

leveraged to analyze the privacy cost in the training process of denoiser. We also note a special case of **MGM**, in which the covariance matrix of the multivariate Gaussian perturbation only contains the diagonal values, and the perturbation on each elements is independent from each other but they can have different scales. This mechanism is called Heterogeneous Gaussian Mechanism (**HGM**) [72]. We re-define **HGM** in Appendix .7 and prove that **MGM** is a generalization of **HGM**.

6.2.3 TransDenoiser Training Algorithm

As introduced in Section 6.2.2, the DP guarantee provided by transformed gradient perturbation depends on the transformation matrix and the scale of input noise. In some scenarios, this transformed gradient perturbation itself does not fully satisfy the DP requirement, because the scale of input noise to achieve randomized smoothing of certified robustness is relatively small. To address this, we add additional gradient perturbation directly to the gradient in each iteration of the training process.

Algorithm 3 shows the details of our proposed TransDenoiser training algorithm to achieve both certified robustness and DP. Each record is perturbed with input perturbation to achieve certified robustness (line 7). We utilize both input perturbation and gradient perturbation to achieve DP for "non-negative cases" (Line 12 - 18), and for the "negative cases", we directly employ gradient perturbation (Line 19 - 23). For non-negative cases, we transform input perturbation at each iteration into gradient perturbation. We then add additional gradient perturbation with scale $\bar{\sigma}$ to $\nabla_{\theta_t} l(\mathbf{z}_t, \theta_t)$ at each iteration given hyper-parameters ξ_{low} and ξ_{up}. In Algorithm 3, we use the commonly used approach, mini-batch SGD, to train the denoiser, which is slightly different from our previous setting (Lemma 1) where only a single sample is fed into the model per iteration. We will show that our DP analysis works in both of these two settings.

Privacy Analysis. In order to compose the transformed gradient perturbation and the direct isotropic gradient perturbation in each iteration for DP analysis, we introduce Mixed

Algorithm 3: TransDenoiser Training Algorithm

Input: pre-trained classifier h, total training epoch T, perturbation scale thresholds ξ_{low} and ξ_{up}, input perturbation scale σ, learning rate η, training dataset D_{train}

t = 0;
load parameters of the pre-trained classifier h;
initialize parameters of the denoiser g;
while $t < T$ **do**
  get mini-batch data $\mathbf{x}_t$ from D_{train};
  for *each data* $\mathbf{x}_{(i)}$ *in* $\mathbf{x}_t$ **do**
    $\mathbf{z}_{(i)} := \mathbf{x}_{(i)} + \mathcal{N}(0, \sigma^2)$;
    $o(\mathbf{z}_{(i)} - \mathbf{x}_{(i)}) := l(\mathbf{z}_{(i)}, \theta) - (l(\mathbf{x}_{(i)}, \theta) + (\mathbf{z}_{(i)} - \mathbf{x}_{(i)})^\intercal \nabla_{\mathbf{x}_{(i)}} l(\mathbf{x}_{(i)}, \theta))$;
  end
  "Non-negative cases" $\mathbf{z}_t^{non} := \{\mathbf{z}_{(i)}\}$ with $o(\mathbf{z}_{(i)} - \mathbf{x}_{(i)}) \geq 0$;
  "Negative cases" $\mathbf{z}_t^{neg} := \{\mathbf{z}_{(i)}\}$ with $o(\mathbf{z}_{(i)} - \mathbf{x}_{(i)}) < 0$;
  if *"Non-negative cases"* **then**
    compute the gradient $\nabla_{\theta_t} l(\theta_t, \mathbf{z}_t^{non})$;
    calculate the input perturbation transformation matrix $\mathbf{M}_t$;
    calculate $S_{min}(\mathbf{M}_t)$;
    $$\bar{\sigma} := \begin{cases} \xi_{up}, & \text{if } \sqrt{TS_{min}(\mathbf{M}_t)}\sigma < \xi_{low}, \\ \sqrt{\xi_{up}^2 - TS_{min}(\mathbf{M}_t)\sigma^2}, & \text{else if } \sqrt{TS_{min}(\mathbf{M}_t)}\sigma < \xi_{up}, ; \\ 0, & \text{else}, \end{cases}$$
    add additional perturbation to the gradient
    $\nabla_\theta l(\theta_t, \mathbf{z}_t^{non}) := \nabla_\theta l(\theta_t, \mathbf{z}_t^{non}) + \mathcal{N}(0, \bar{\sigma}^2)$;
  end
  else
    compute the gradient $\nabla_{\theta_t} l(\theta_t, \mathbf{z}_t^{neg})$;
    $\bar{\sigma} := \xi_{up}$;
    add perturbation to the gradient $\nabla_\theta l(\theta_t, \mathbf{z}_t^{neg}) := \nabla_\theta l(\theta_t, \mathbf{z}_t^{neg}) + \mathcal{N}(0, \bar{\sigma}^2)$;
  end
  $\nabla_{\theta_t} l(\theta_t, \mathbf{z}_t) := \nabla_\theta l(\theta_t, \mathbf{z}_t^{non}) + \nabla_\theta l(\theta_t, \mathbf{z}_t^{neg})$;
  update parameter for next iteration
  $\theta_{t+1} := \theta_t - \eta \frac{1}{B} \sum_{i=1}^{B} (\nabla_\theta l(\theta_t, \mathbf{z}_t^{non}) + \nabla_\theta l(\theta_t, \mathbf{z}_t^{neg}))$;
end
output θ_T and compute overall privacy cost through moments accountant.

Multivariate Gaussian Analysis below.

Theorem 4. *Mixed Multivariate Gaussian Analysis. Let* $\mathcal{G} : \mathbb{R}^v \rightarrow \mathbb{R}^w$ *be an arbitrary* w*-dimensional function, and* $\Delta_{\mathcal{G}} = \max_{\mathcal{D},\mathcal{D}'} \|\mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')\|_2$*. Mixed Multivariate Gaussian Analysis is the mix of a Multivariate Gaussian Mechanism* $\mathcal{M}_1$ *with the covariance* $\mathbf{\Sigma}_{(i)} \in \mathbb{R}^{w \times w}$ *and a Gaussian Mechanism* $\mathcal{M}_2$ *with* $\bar{\sigma}$ *adding noise to each of the* w *elements of the output. This mixed mechanism is* (ϵ, δ)*-DP, with*

$$\epsilon \in (0, 1], \sqrt{\bar{\sigma}^2 + TS_{min}(\mathbf{M}_{(i)})\sigma^2} \geq \xi_{up} \geq \sqrt{2\ln(1.25/\delta)}\Delta_{\mathcal{G}}/\epsilon.$$

where $\bar{\sigma}$ *is the isotropic gradient perturbation,* T *is the number of training steps,* $S_{min}(\mathbf{M}_{(i)})$ *is the minimum singular value of* $\mathbf{M}_{(i)}$ *and* $\mathbf{\Sigma}_{(i)} \triangleq \mathbf{M}_{(i)}\sigma^2$*.*

Mixed Multivariate Gaussian Analysis (**MMGA**) can be leveraged to analyze privacy guarantee provided by transformed gradient perturbation and additional gradient perturbation. In the following of this section, we will prove Theorem 4 and show that **MMGA** can work with moments accountant to provide a tighter DP estimation and preserve (ϵ, δ)-DP for deep learning models with pre-trained classifiers.

Privacy Analysis for Vanilla SGD. In vanilla SGD, the algorithm randomly picks one sample at each iteration and feeds it into the model for optimization. Given the initial parameters θ_0, iteration t, the parameters are updated as: $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} l(\theta_t, \mathbf{z}_t^{non})$, where η denotes the learning rate and $\mathbf{z}_t^{non}$ denotes one perturbed sample randomly picked at iteration t. In optimization process, the transformed perturbation $\mathbf{p}_t$ is slightly different from the that in Equation (6.3).

Lemma 2. *Given perturbed example* $\mathbf{z}_t^{non} = \mathbf{x}_t^{non} + \mathbf{b}_t$ *with* $\mathbf{b}_t^{(k)} \sim \mathcal{N}(0, \sigma^2)$*, the number of training steps* T*, and* C*-Lipschitz continuous loss* l*. The gradient* $\nabla_{\theta_t} l(\mathbf{z}_t^{non}, \theta_t)$ *at each step of vanilla SGD can be reformulated as:*

$$\nabla_{\theta_t} l(\mathbf{z}_t^{non}, \theta_t) = \nabla_{\theta_t} l(\mathbf{x}_t^{non}, \theta_t) + \mathbf{p}_t, \tag{6.4}$$

where the transformed perturbation $\mathbf{p}_t \sim \mathcal{N}(0, TS_{min}(\mathbf{M}_t)\sigma^2\mathbf{I})$, $\mathbf{M}_t = \mathbf{A}_{(i)}\mathbf{A}_{(i)}^\intercal$, $\mathbf{A}_{(i)} = \mathbf{J}_{\theta_t}\nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta_t)$.

The detailed proof of Lemma 2 can be found in Appendix .8. Theorem 4 can be proved by Lemma 2, the definition of $\bar{\sigma}$ and Theorem 3. Thus, we have the following corollary for "non-negative cases":

Corollary 2. *Given an input perturbation* $\mathbf{b}_t \in \mathbb{R}^v$ *with* $\mathbf{b}_t^{(k)} \sim \mathcal{N}(0, \sigma^2)$*, the transformation matrix* $\mathbf{A}_t$ *and additional gradient perturbation with scale* $\bar{\sigma}$*, the DP guarantee of the* ***MMGA*** *is equivalent to that of a* ***GM*** *with its perturbation following a Gaussian distribution* $\mathcal{N}(0, \xi_{up}^2)$.

The above analysis for "non-negative cases" leverages perturbation transformation and **MMGA** to analyze DP. For "negative cases", because the perturbation transformation is no longer required and the gradient perturbation with $\bar{\sigma} = \xi_{up}$ is directly added to $\nabla_\theta l(\theta_t, \mathbf{z}_t^{neg})$, the **MMGA** is equivalent to traditional **GM** with perturbation following $\mathcal{N}(0, \xi_{up}^2)$. Therefore, we can conclude that Corollary 2 is applicable to both "non-negative cases" and "negative cases".

Privacy Analysis for Mini-batch SGD. The above claims for Vanilla SGD can be adapted to Mini-batch SGD by setting $\mathbf{M}_t = \frac{1}{B^2}\sum_{i=1}^{B}\mathbf{A}_{(i)}\mathbf{A}_{(i)}^\intercal$ instead of $\mathbf{M}_t = \mathbf{A}_{(i)}\mathbf{A}_{(i)}^\intercal$ in Vanilla SGD. The detailed proof can be found in Appendix .9.

Tighter Composition via Moments accountant. While Corollary 2 is derived with simple compositions, moments accountant can be applied with **MMGA** to provide a tighter DP composition for Algorithm 3.

Theorem 5. *There exist constants* c_1 *and* c_2 *so that given sampling probability* $q = \frac{B}{N}$ *and the number of training steps* T*, for any* $\epsilon < c_1 q^2$*, Algorithm 3 is* (ϵ, δ)*-differential private for any* $\delta > 0$ *if*

$$\xi_{up} \geq c_2 \frac{q\sqrt{Tlog(1/\delta)}}{\epsilon} \tag{6.5}$$

The proof of Theorem 9 can be found in Appendix .10.

Discussion. We note that although the transformation matrix $\mathbf{A}_{(i)}$ requires the clean example $\mathbf{x}_{(i)}$, the calculation of $\mathbf{A}_{(i)}$ does not incur privacy cost for the denoiser. This is because the transformation process and the calculation of $\mathbf{A}_{(i)}$ is only for analyzing the DP of the input perturbation, i.e., the clean example $\mathbf{x}_{(i)}$ does not actually contribute to the gradient $\nabla_\theta l(\mathbf{z}_{(i)}^{non}, \theta)$. Another potential privacy concern is about the observations that "non-negative cases" and "negative cases" use different scales of additional gradient perturbation. Even for "non-negative cases", different data will be applied with different $\bar{\sigma}$. We claim that the different additional perturbation scales among "non-negative cases" and "negative cases" will not incur privacy violation, because we have proven that input perturbation also provides certain level of privacy guarantee, which is analyzed in a way that we transform it into gradient perturbation. We first use perturbation transform matrix and **MGM** to calculate how much gradient perturbation scale can be transformed from input perturbation for different data. Then we add additional gradient perturbation onto transformed gradient perturbation to ensure that the overall gradient perturbation scale $\bar{\sigma}^2 + TS_{min}(\mathbf{M}_t)\sigma^2 \geq \xi_{up}^2$ for each data. On the one hand, the calculation of $\bar{\sigma}$ is not visible to users or attackers. On the other hand, different $\bar{\sigma}$ can ensure the overall scale $\bar{\sigma}^2 + TS_{min}(\mathbf{M}_t)\sigma^2$ is a consistent lower bound for all training data.

6.3 Experiments

In this section, we will show our experiments on two benchmark datasets, MNIST and CIFAR-10. These experiments are conducted to prove that 1) TransDenoiser can provide high level of certified robustness through randomized smoothing, 2) the input perturbation transformation can save a considerable amount of DP budget and thus improve the model performance.

6.3.1 Configurations

Baseline and ablation studies. We employ SecureSGD [72] and StoBatch [71] two architectures in baseline approaches, and compare the certified robustness and DP performance on MNIST and CIFAR-10 datasets. We acquire two versions of SecureSGD through different training strategies: SecureSGD_sct is acquired by training an entire classifier from scratch, and SecureSGD_prt is acquired by training the denoiser and fixing the pre-trained classifier. For StoBatch, we only acquire the one training from scratch, because the denoiser with pre-trainied classifier can not fit to its architecture. We also conduct two ablation studies with two variants of TransDenoiser: 1) TransDenoiser_nodp which only contains input perturbation for certified robustness, without the perturbation transformation and additional gradient perturbation for DP; 2) TransDenoiser_sepdp which contains input perturbation for certified robustness and separate gradient perturbation for DP, without utilizing input perturbation via perturbation transformation.

Models. Pre-trained classifiers are trained on public datasets, and we use convolutional and transposed convolutional layers to build the autoencoder based denoiser for both MNIST and CIFAR10 datasets. The details of pre-trained classifiers and denoiser can be found in Appendix .14.

Adversarial examples. We use four different attack algorithms, i.e., FGSM, I-FGSM [52], Momentum Iterative Method (MIM) [24], and MadryEtAl [64], to craft adversarial examples. These algorithms apply l_2-norm attack on the pre-trained classifier under a white-box setting. Given a threshold L_{atk} of perturbation norm, adversarial example $\mathbf{x}'$ can be represented as $\mathbf{x}' = \mathbf{x} + \psi$ s.t. $\forall \psi \in \mathcal{R}^v, \|\psi\|_2 \leq L_{atk}$.

Certification. We employ the randomized smoothing of Cohen et al. [21] in our work. A function $robustRadius(\mathbf{x}_{(i)}, \sigma)$ is designed to return a certified radius κ given the input and its perturbation scale. This indicates that the randomized smoothed model is certified robust around $\mathbf{x}_{(i)}$ within the radius κ.

Evaluation metrics. We evaluate the performance in terms of certified accuracy (CertAcc) on clean examples and conventional accuracy (ConvAcc) on both clean and adversarial examples. $CertAcc = \frac{1}{N} isCorrect(\mathbf{x}_{(i)})(robustRadius(\mathbf{x}_{(i)}, \sigma) \geq R)$, $ConvAcc = \frac{isCorrect(\mathbf{x}'_{(i)})}{N}$ for adversarial example; $\frac{isCorrect(\mathbf{x}_{(i)})}{N}$ for clean example, where N denotes the test data size, $\mathbf{x}'_{(i)}$ denotes the adversarial example, $isCorrect(\mathbf{x}_{(i)})$ denotes the function returning 1 when the prediction on $\mathbf{x}_{(i)}$ is correct and 0 otherwise, $isCorrect(\mathbf{x}'_{(i)})$ works the same on $\mathbf{x}'_{(i)}$, $robustRadius(\mathbf{x}_{(i)}, \sigma) \geq R$ returns 1 when the certified radius κ is equal or larger than the threshold R and 0 otherwise.

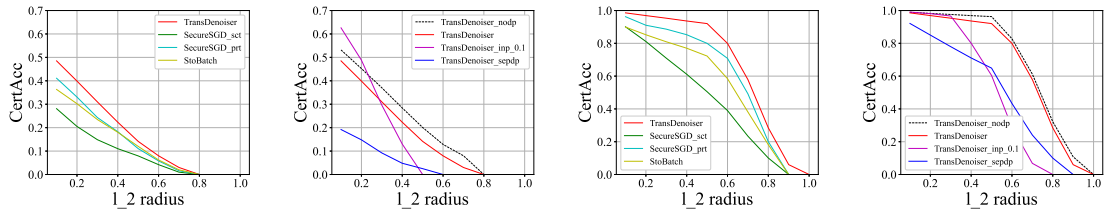
Implementation details. The detailed implementation and code can be found in Appendix .14.

6.3.2 Experimental Results

We conduct our experiments on MNIST and CIFAR-10 to show that TransDenoiser can simultaneously achieve both differential privacy and certified robustness via input and gradient perturbation.

For the following experiments, we will compare with 1) SecureSGD_sct, SecureSGD_prt and StoBatch to show that TransDenoiser achieves better performance than baselines on certified robustness and DP, 2) compare with TransDenoiser_nodp to show that TransDenoiser still achieves high level of certified robustness after adding gradient perturbation for DP, and 3) compare with TransDenoiser_sepdp to show that input perturbation transformation effectively saves a considerable amount of DP budget and improves the utility performance.

Certified robustness. We demonstrate that TransDenoiser can achieve high level of certified robustness on both MNIST and CIFAR10. We conduct experiments to measure the CertAcc on clean examples with different l_2 radii of different methods on the two datasets, shown in Figure 6.2a and Figure 6.2c. TransDenoiser significantly outperforms the state-of-the-art SecureSGD_prt, SecureSGD_sct and StoBatch algorithms on both datasets, thanks



(a) TransDenoiser and baselines on CIFAR10 (b) TransDenoiser and ablation cases on CIFAR10 (c) TransDenoiser and baselines on MNIST (d) TransDenoiser and ablation cases on MNIST

Figure 6.2: Comparison among TransDenoiser, baselines and ablation cases for certified accuracy vs. l_2 radii on two datasets. The input perturbation scale = 0.25, the overall gradient perturbation scale = 2.0 ($\geq$ 2.0 for TransDenoiser), and guarantee $(1.0, 1\mathrm{e}{-5})$-DP for private models.

to the benefit of perturbation transformation and randomized smoothing.

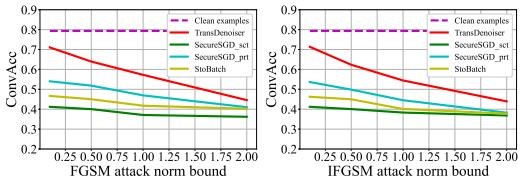
Figure 6.2b and Figure 6.2d show the comparison with ablation cases. Besides TransDenoiser_nodp and TransDenoiser_sepdp that we already introduced in Sec 6.3.1, we add one more ablation case TransDenoiser_inp_0.1 denoting that the input perturbation scale for this TransDenoiser is 0.1 rather than 0.25. Comparing these ablation cases, TransDenoiser achieves similar CertAcc as TransDenoiser_nodp, which means that TransDenoiser needs to add very little additional perturbation for ensuring DP thanks to the benefit of input perturbation that is being exploited. Compared with TransDenoiser_sepdp that uses separate gradient perturbation for DP, TransDenoiser saves significant DP budget and thus requires less perturbation for DP, leading to significantly higher accuracy. Comparing with TransDenoiser_inp_0.1, we find that TransDenoiser_inp_0.1 can achieve highest CertAcc when l_2 radius is small, but it drops quickly as radius increases. This is because 1) smaller input perturbation scale will bring less randomization to the model, and thus improves performance; 2) smaller input perturbation scale can only defend against less "powerful" adversarial attacks, and thus CertAcc drops when attack radius increases. Comparing the two datasets, they show similar trend besides the fact that MNIST has higher accuracy for all methods in general due to its simplicity.

Empirical defense. Certified robustness shows the theoretical defense against adversarial examples, we also conduct experiments to show that TransDenoiser can empirically defend against adversarial examples from different attacks. We only show the results against

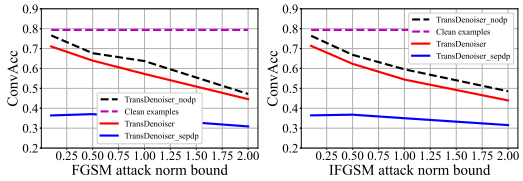
FGSM, I-FGSM attacks here, the more detailed experiments can be found in Appendix .14. Figure 9a and Figure 9c show the convAcc of TransDenoiser and baselines with respect to varying attack norm bound for different attack methods on two datasets, respectively. Compared with these baselines, TransDenoiser achieves better empirical performance with any attack norm bound of all attacks. Figure 9b and Figure 9d show the comparison of TransDenoiser and ablation cases. Compared with TransDenoiser_nodp, TransDenoiser achieves similar ConvAcc, which proves that perturbation for DP in TransDenoiser does not affect the ConvAcc too much. Compared with TransDenoiser_sepdp, TransDenoiser effectively saves DP budget and achieves better empirical performance against adversarial examples. In all these figures, we add a curve named "Clean examples" to represent ConvAcc that clean examples pass through TransDenoiser. As can be seen, ConvAcc for clean examples keep consistent as attack norm bound increasing, and TransDenoisercan achieve similar ConAcc as "Clean examples" when attack norm bound is small.

Observing the similar results between CertAcc and ConvAcc, we see that the certified accuracy on clean examples provides a good estimation for the empirical robustness of the model. If a model achieves relatively high CertAcc on clean examples, it can have a high probability to achieve high ConvAcc on adversarial examples.

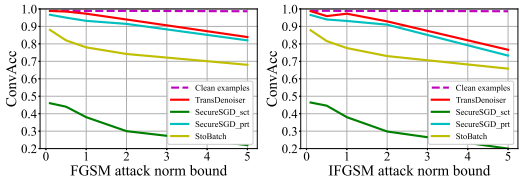
Differential privacy. We also evaluate the tradeoff between accuracy and privacy for different methods. As can be seen from Figure 6.4a and Figure 6.4b, given the same ϵ, TransDenoiser can always achieve higher ConvAcc on different attacks similar to what we have observed so far. In addition, with increasing epsilon, all methods achive a higher accuracy as expected. On MNIST dataset, TransDenoiser_sepdp achieves similar result with TransDenoiser.



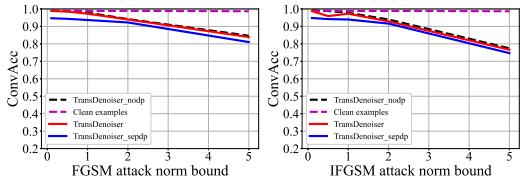
(a) TransDenoiser and baselines on CIFAR10



(b) TransDenoiser and ablation cases on CIFAR10

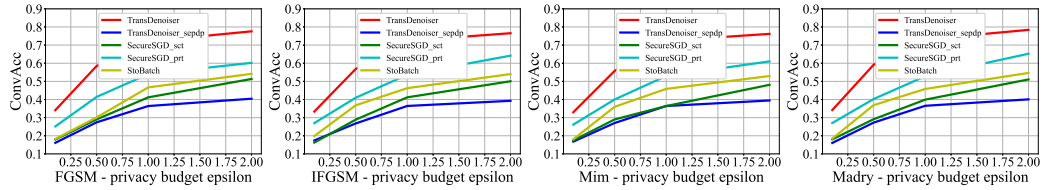


(c) TransDenoiser and baselines on MNIST

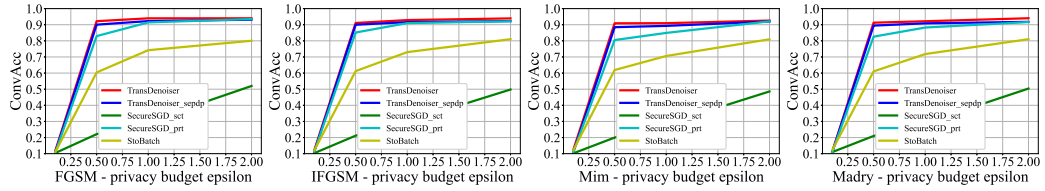


(d) TransDenoiser and ablation cases on MNIST

Figure 6.3: More comparison among TransDenoiser, baselines and ablation cases for conventional accuracy vs. l_2 radii on two datasets. The input perturbation scale on CIFAR10 = 0.1, on MNIST = 0.25, the overall gradient perturbation scale = 2.0 ($\geq$ 2.0 for TransDenoiser), and guarantee $(1.0, 1\text{e}{-5})$-DP for private models.



(a) TransDenoiser, baselines and ablation cases on CIFAR10



(b) TransDenoiser, baselines and ablation cases on MNIST

Figure 6.4: Comparison among TransDenoiser, baselines and ablation cases for conventional Accuracy vs. ϵ on two datasets. The input perturbation scale on CIFAR10 = 0.1, on MNIST = 0.25, attack norm bound = 2.0, the overall gradient perturbation scale = 2.0 ($\geq$ 2.0 for TransDenoiser), and $\delta = 1\text{e}{-5}$ for DP.

Chapter 7

Conclusions and Future Work

In Chapter 4, we have proposed a new framework IGAMT to generate differentially private EHRs with heterogeneous features, missing values and irregular measures. IGAMT leverages missing value masks and sequence-to-sequence transformers with well-designed embeddings to learn the underlying characteristics of EHRs, and generate synthetic data of high quality. By leveraging the elaborate architecture and objective function, imitator of IGAMT is capable to imitate the behaviors of the decoder and learn from different sources of EHRs while reducing randomization brought by DP. After training with gradient perturbation, IGAMT will release G, Dec_i and the last layer of Dec as a differentially private generative model. We conduct a large volume of experiments to prove that IGAMT achieves the state-of-the-art performance on DP EHRs synthesization. In Chapter 5, we proposed a Distance Constrained Adversarial Imitation Network AIN to generate both targeted and untargeted adversarial examples. By incorporating imitation based learning from existing adversarial examples by optimization based methods, AIN combines the benefit of both model based methods and the traditional optimization based methods. Our experiments on two benchmark datasets verified the performance advantage of the proposed approach. AIN can achieve higher attack success rate with smaller perturbation distances than the state-of-the-art model based methods ATN and AdvGAN, and similar performance

to the state-of-the-art optimization based methods such as i-FGSM and C&W. In Chapter 6, we have proposed TransDenoiser to achieve both DP and certified robustness via input perturbation. TransDenoiser stands as the first attempt to achieve both for the vastly existing, yet under-studied, pre-trained model setting. We leverage input perturbation transformation to efficiently transform input perturbation into gradient perturbation. We propose **MGM** and **MMGA** to analyze DP of the transformed gradient perturbation and combine **MMGA** with moments accountant to provide a tight bound on DP guarantee. Therefore, TransDenoiser effectively saves a considerable DP budget and improves the utility performance compared to using gradient perturbation independently to achieve DP. Our experiments on two benchmark datasets verify the performance advantage of TransDenoiser w.r.t. both DP and certified robustness compared to state-of-the-art methods.

In deep learning area, the utility of DP models are typically limited by the randomization. Generative model with our proposed architecture, imitator, could be one of the potential direction to achieve better trade-off between utility and privacy. In the future work, we will explore advanced architectures that can facilitate the utility/privacy performance. For AIN, it would be interesting to train AIN in the black-box manner by model stealing [84] and analyze its performance. Applying the idea of imitation to a GAN-based method [87] can also be a potential direction of study. For achieving both differential privacy and certified robustness, we have build a bridge between input perturbation and gradient perturbation in deep learning models, and transform from the input perturbation side to the gradient perturbation side. It would be meaningful to explore whether we could transform in the opposite direction and, moreover, to explore whether a DP model can be quantitatively analyzed to defend against adversarial examples.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] 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*, pages 308–318, 2016.

[3] Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Nicolas Papernot, Kunal Talwar, and Li Zhang. On the protection of private information in machine learning systems: Two recent approches. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 1–6. IEEE, 2017.

[4] Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

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

[6] Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[8] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Thirty-second aaai conference on artificial intelligence*, 2018.

[9] Seo-Jin Bang, Yuchuan Wang, and Yang Yang. Phased-lstm based predictive model for longitudinal ehr data with missing values, 2020.

[10] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 2019.

[11] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Thakurta. Private stochastic convex optimization with optimal rates. *arXiv preprint arXiv:1908.09970*, 2019.

[12] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[13] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

[14] Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. In *Theory of Cryptography Conference*, pages 437–454. Springer, 2010.

[15] Joan Bruna, Christian Szegedy, Ilya Sutskever, Ian Goodfellow, Wojciech Zaremba, Rob Fergus, and Dumitru Erhan. Intriguing properties of neural networks. 2013.

[16] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[17] Thee Chanyaswad, Alex Dytso, H Vincent Poor, and Prateek Mittal. Mvg mechanism: Differential privacy under matrix-valued query. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 230–246, 2018.

[18] Kieran Chin-Cheong, Thomas Sutter, and Julia E Vogt. Generation of differentially private heterogeneous electronic health records. *arXiv preprint arXiv:2006.03423*, 2020.

[19] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[20] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[21] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[22] Stanford CS231N. Tiny imagenet visual recognition challenge. `https://tiny-imagenet.herokuapp.com/`, 2017.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[24] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Discovering adversarial examples with momentum. *arXiv preprint arXiv:1710.06081*, 2017.

[25] Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[26] 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.

[27] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[28] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

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

[30] Vitaly Feldman, Tomer Koren, and Kunal Talwar. Private stochastic convex optimization: optimal rates in linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 439–449, 2020.

[31] James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[32] 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.

[33] 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 *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 151–164. Springer, 2019.

[34] Kazuto Fukuchi, Quang Khai Tran, and Jun Sakuma. Differentially private empirical risk minimization with input perturbation. In *International Conference on Discovery Science*, pages 82–90. Springer, 2017.

[35] Manuel Gil. *On Rényi divergence measures for continuous alphabet sources*. PhD thesis, Citeseer, 2011.

[36] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[38] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[39] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[42] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, 6:3–10, 1994.

[43] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*, 2017.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[45] Stephanie Hyland, Cristóbal Esteban, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. 2018.

[46] Alistair Johnson et al. Alistair johnson, lucas bulgarelli, tom pollard, leo anthony celi, roger mark, steven horng.

[47] Yilin Kang, Yong Liu, Lizhong Ding, Xinwang Liu, Xinyi Tong, and Weiping Wang. Differentially private erm based on data perturbation. *arXiv preprint arXiv:2002.08578*, 2020.

[48] Yilin Kang, Yong Liu, Ben Niu, Xinyi Tong, Likun Zhang, and Weiping Wang. Input perturbation: A new paradigm between central and local differential privacy. *arXiv preprint arXiv:2002.08570*, 2020.

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

[50] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[52] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[53] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[54] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[55] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[56] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[57] Dongha Lee, Hwanjo Yu, Xiaoqian Jiang, Deevakar Rogith, Meghana Gudala, Mubeen Tejani, Qiuchen Zhang, and Li Xiong. Generating sequential electronic health records using dual adversarial autoencoder. *Journal of the American Medical Informatics Association*, 27(9):1411–1419, 2020.

[58] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi Jaakkola. Tight certificates of adversarial robustness for randomly smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 4910–4921, 2019.

[59] Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1656–1665, 2018.

[60] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2018.

[61] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, pages 9464–9474, 2019.

[62] Yikuan Li, Shishir Rao, Jose Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. Behrt: transformer for electronic health records. *Scientific reports*, 10(1):1–12, 2020.

[63] Xinhai Liu, Zhizhong Han, Xin Wen, Yu-Shen Liu, and Matthias Zwicker. L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 989–997. ACM, 2019.

[64] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[65] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[66] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[67] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[68] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *arXiv preprint arXiv:1610.09513*, 2016.

[69] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[70] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[71] Hai Phan, My T Thai, Han Hu, Ruoming Jin, Tong Sun, and Dejing Dou. Scalable differential privacy with certified robustness in adversarial learning. In *International Conference on Machine Learning*, pages 7683–7694. PMLR, 2020.

[72] NhatHai Phan, Minh Vu, Yang Liu, Ruoming Jin, Dejing Dou, Xintao Wu, and My T Thai. Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness. *arXiv preprint arXiv:1906.01444*, 2019.

[73] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4322–4330, 2019.

[74] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. *arXiv preprint arXiv:2003.01908*, 2020.

[75] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019.

[76] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[77] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[78] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[79] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[80] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, pages 8312–8323, 2018.

[81] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[82] 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.

[83] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[84] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[85] 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*, pages 5998–6008, 2017.

[86] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[87] Xiaosen Wang, Kun He, Chuan Guo, Kilian Q Weinberger, and John E Hopcroft. At-gan: A generative attack model for adversarial transferring on generative adversarial nets. *arXiv preprint arXiv:1904.07793*, 2019.

[88] Yue Wang, Cheng Si, and Xintao Wu. Regression model fitting under differential privacy and model inversion attack. In *IJCAI*, pages 1003–1009, 2015.

[89] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[90] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[91] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[92] Chugui Xu, Ju Ren, Deyu Zhang, Yaoxue Zhang, Zhan Qin, and Kui Ren. Ganobfuscator: Mitigating information leakage under gan via differential privacy. *IEEE Transactions on Information Forensics and Security*, 14(9):2358–2371, 2019.

[93] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *arXiv preprint arXiv:1907.00503*, 2019.

[94] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016.

[95] Jungang Yang, Liyao Xiang, Jiahao Yu, Xinbing Wang, Bin Guo, Zhetao Li, and Baochun Li. Matrix gaussian mechanisms for differentially-private learning. *IEEE Transactions on Mobile Computing*, 2021.

[96] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349. IEEE, 2019.

[97] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[98] Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. Walking on the edge: Fast, low-distortion adversarial examples. *IEEE Transactions on Information Forensics and Security*, 16:701–713, 2020.

[99] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: regression analysis under differential privacy. *arXiv preprint arXiv:1208.0219*, 2012.

[100] Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594*, 2018.

[101] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5810–5818, 2017.

Appendices

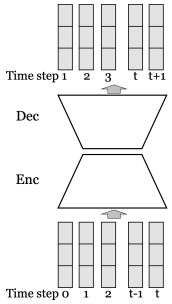


Figure 1: An illustration of sequence-to-sequence autoencoder.

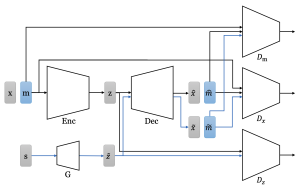


Figure 2: The framework of EDGAMT.

.1 Details of Architecture

Seq2seq AE. Figure 1 shows the details of sequence-to-sequence autoencoder.

EDGAMT. $IGAMT\_w/o\_imi\_var$ is also named as EDGAMT, which is a variant of $IGAMT\_w/o\_imi$. It adds an auxiliary discriminator D_m to facilitate the discrimination of masks. In experiments, we find real mask is quite trivial to be differentiated from synthetic masks by D_x, which degrades the quality of synthetic EHRs. To solve this challenge, an extra discriminator D_m is added to discriminate $\mathbf{m}$ from $\hat{\mathbf{m}}$, $\tilde{\mathbf{m}}$ and $\bar{\mathbf{m}}$. Gradient perturbations to achieve DP will be added on gradients of D_x, D_m, D_z and the last layer of Dec. EDGAMT adds an auxiliary discriminator D_m to facilitate the discrimination of masks. Since D_m provides extra loss, the discriminate loss on Imi and Dec differs from that of $IGAMT\_w/o\_imi$. The discriminate loss for Dec is as follows:

$$l_{de} = -(D_x(\hat{\mathbf{x}} \times \hat{\mathbf{m}}) + D_x(\tilde{\mathbf{x}} \times \tilde{\mathbf{m}})) - (D_m(\hat{\mathbf{m}}) + D_m(\tilde{\mathbf{m}})),$$

and the loss for imitator Imi is as follows:

$$l_{di} = -D_x(\bar{\mathbf{x}} \times \bar{\mathbf{m}}) - D_m(\bar{\mathbf{m}}).$$

The extra discriminate loss for D_m is as follows,

$$l_{dm} = (D_m(\hat{\mathbf{m}}) + D_m(\tilde{\mathbf{m}}) + D_m(\bar{\mathbf{m}})) - 3D_m(\mathbf{m}).$$

The other parts of objective function are the same as $IGAMT\_w/o\_imi$. We claim that for IGAMT, EDGAMT is an alternative module to $IGAMT\_w/o\_imi$ and IGAMT can be crafted either on EDGAMT or $IGAMT\_w/o\_imi$. In our experiments, we build IGAMT on EDGAMT.

.2 More Experiments of Work 1

We will show experiment results of Emory Synergy EHRs in this section.

PCA Results. Figure 3 shows the results baselines, and Figure 4 shows the results from ablation cases and IGAMT.

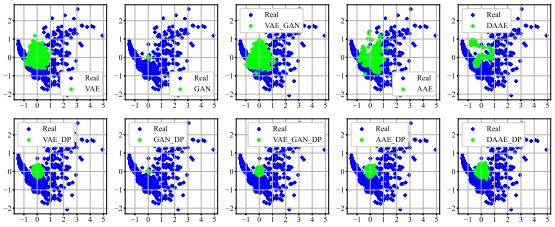


Figure 3: PCA results for real EHRs and synthetic EHRs of baselines.

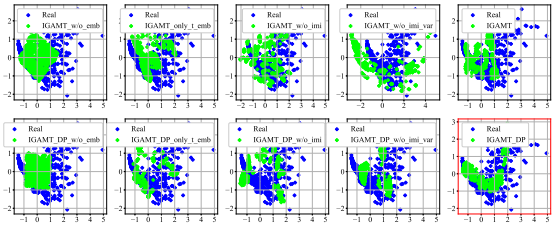


Figure 4: PCA results for real EHRs and synthetic EHRs of ablation cases and IGAMT.

Table 1: Overall similarity with $\sigma = 1.0$.

Model	Minkowski Distance	Cosine Similarity
VAE_{DP}	7.330084	-0.021758
GAN_{DP}	7.009576	-0.057234
VAE_GAN_{DP}	7.205729	0.210885
AAE_{DP}	6.995771	0.183082
$DAAE_{DP}$	6.439933	0.390111
$IGAMT\_w/o\_emb_{DP}$	6.470049	0.396667
$IGAMT\_only\_t\_emb_{DP}$	6.447262	0.389958
$IGAMT\_w/o\_imi_{DP}$	6.174218	0.409063
$IGAMT\_w/o\_imi\_var_{DP}$	6.151663	0.420291
$IGAMT_{DP}$	**5.580912**	**0.560559**

Similarity Comparison. Table 1 shows the overall Minkowski distance and cosine similarity comparison among different models with DP.

Differential Privacy. Figure 5 and Figure 6 show the overall similarity of different models with different privacy budget ϵ and perturbation standard deviation σ respectively.

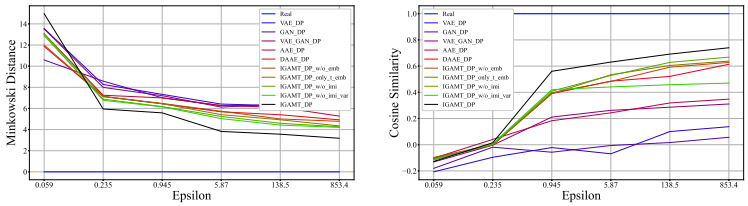


Figure 5: Overall Minkowski distance and cosine similarity on different models with different ϵ.

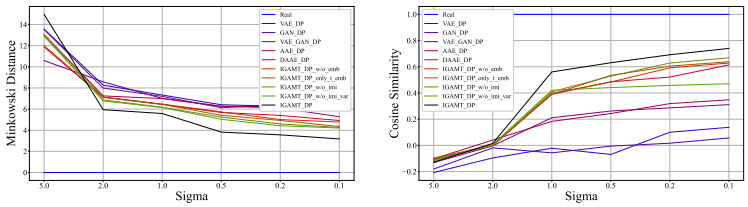


Figure 6: Overall Minkowski distance and cosine similarity on different models with different σ.

.3 Brief Proof of Randomized Smoothing

Given denoiser g and pre-trained classifier h, let $f = g \circ h$, a randomized smoothed classifier $\tilde{f}$ works as follows,

$$\tilde{f}(\mathbf{x}) = \underset{j=1,\ldots,c}{\operatorname{argmax}} Pr[f(\mathbf{x} + \mathbf{b}) = j], \tag{1}$$

where input perturbation $\mathbf{b} \sim \mathcal{N}(0, \sigma^2 I)$.

Suppose the probability that $\tilde{f}$ return the class y of $\mathbf{x}$ is $p_y = \mathcal{P}(\tilde{f}(\mathbf{x}) = y)$ and the probability for "runner-up" class b is $p_b = \mathcal{P}(\tilde{f}(\mathbf{x}) = b)$. The smoothed classifier $\tilde{f}$ is certified robust around $\mathbf{x}$ within the radius

$$\kappa = \frac{\sigma}{2}(\Phi^{-1}(p_y) - \Phi^{-1}(p_b)),$$

where Φ^{-1} is the inverse of the standard Gaussian CDF. However, computing p_y and p_b is not practical for deep learning models. To solve this challenge, Cohen et al. [21] used Monte Carlo sampling to get the estimation $\underline{p_y}$ and $\overline{p_b}$ s.t. $\underline{p_y} \leq p_y$ and $\overline{p_b} \geq p_b$ with arbitrarily high probability. $\underline{p_y}$ and $\overline{p_b}$ is then used to substitute p_y and p_b in Equation (1) and the certified radius κ is obtained.

.4 Theorem 6 (wEGM) and Proof

w-Element Gaussian Mechanism (**wEGM**) is actually a rewrite of the traditional Gaussian mechanism and can guarantee that $\mathcal{M}$ is still DP after applying perturbation to each component of the output.

Theorem 6. *w-Element Gaussian Mechanism. Let $\mathcal{G} : \mathbb{R}^v \to \mathbb{R}^w$ be an arbitrary w-dimensional function, and $\Delta_{\mathcal{G}} = \max_{\mathcal{D},\mathcal{D}'} \|\mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')\|_2$. A w-Element Gaussian Mechanism $\mathcal{M}$ with σ adds noise to each of the w elements of the output. The mechanism $\mathcal{M}$ is*

(ϵ, δ)-***DP**, with*

$$\epsilon \in (0, 1],\ \sigma \geq \sqrt{2\ln(1.25/\delta)}\Delta_{\mathcal{G}}/\epsilon.$$

As can be seen in the definition of the theorem, the perturbation in **wEGM** is i.i.d. with the same scale σ. The proof of Theorem 6 is as follows:

Proof. The privacy loss of an output o is defined as:

$$\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}') = ln\frac{Pr[\mathcal{M}(\mathcal{D}, \mathcal{G}, \sigma)] = o}{Pr[\mathcal{M}(\mathcal{D}', \mathcal{G}, \sigma)] = o} \tag{2}$$

Given $v = \mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')$, we have

$$\begin{aligned}
&|\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= |ln\frac{Pr[\mathcal{M}(\mathcal{D}, \mathcal{G}, \sigma)] = o}{Pr[\mathcal{M}(\mathcal{D}', \mathcal{G}, \sigma)] = o}| \\
&= |ln\frac{Pr[\mathcal{G}(\mathcal{D}) + \mathcal{N}(0, \sigma^2)] = o}{Pr[\mathcal{G}(\mathcal{D}') + \mathcal{N}(0, \sigma^2)] = o}| \\
&= |ln\frac{\prod_{k=1}^{w} exp(-\frac{1}{2\sigma^2}(o_k - \mathcal{G}(\mathcal{D})_k)^2}{\prod_{k=1}^{w} exp(-\frac{1}{2\sigma^2}(o_k - \mathcal{G}(\mathcal{D})_k + v_k)^2}| \\
&= \frac{1}{2\sigma^2}|\sum_{k=1}^{w}(o_k - \mathcal{G}(\mathcal{D})_k)^2 - (o_k - \mathcal{G}(\mathcal{D})_k + v_k)^2|.
\end{aligned}$$

Let p denotes $o - \mathcal{G}(\mathcal{D})$, then $p_k = o_k - \mathcal{G}(\mathcal{D})_k$, and $p_k \sim \mathcal{N}(0, \sigma^2)$, Then we have

$$\begin{aligned}
&|\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= \frac{1}{2\sigma^2}|\sum_{k=1}^{w}(p_k^2 - (p_k + v_k)^2)| \\
&= \frac{1}{2\sigma^2}|\|p\|^2 - \|p + v\|^2|.
\end{aligned}$$

According to the fact that a spherically symmetric normal distribution is independent of the orthogonal basis, we can always find a basis $b_1, b_2, ..., b_w$ that is aligned with p. Then we

can depict $p = \sum_{k=1}^{w} {p'}_k$ with ${p'}_k$ being the component on basis b_k and ${p'}_k \sim \mathcal{N}(0, \sigma^2)$. Without loss of generality, we assume b_1 is parallel to v, which means that

$$\|p\|^2 = \sum_{k=1}^{w} \|{p'}_k\|^2,$$
$$\|p + v\|^2 = \|{p'}_1 + v\|^2 + \sum_{k=2}^{w} \|{p'}_k\|^2.$$

Therefore, we have

$$\begin{aligned}
&|\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= \frac{1}{2\sigma^2} |\|p\|^2 - \|p + v\|^2| \\
&= \frac{1}{2\sigma^2} |\|{p'}_1 + v\|^2 - \|{p'}_1\|^2| \\
&\leq \frac{1}{2\sigma^2} |\Delta_{\mathcal{G}}^2 + 2|{p'}_1|\Delta_{\mathcal{G}}|
\end{aligned}$$

Following the proof of Theorem A.1 in [27], we can show that the mechanism $\mathcal{M}$ is (ϵ, δ)-DP, with

$$\epsilon \in (0, 1], \sigma \geq \sqrt{2ln(1.25/\delta)}\Delta_{\mathcal{G}}/\epsilon.$$

□

.5 Proof of Theorem 3 (MGM)

Proof. The privacy loss of an output o is defined as:

$$\mathcal{L}(\mathbf{o}; \mathcal{M}, \mathcal{D}, \mathcal{D}') = ln \frac{Pr[\mathcal{M}(\mathcal{D}, \mathcal{G}, \mathbf{\Sigma})] = \mathbf{o}}{Pr[\mathcal{M}(\mathcal{D}', \mathcal{G}, \mathbf{\Sigma})] = \mathbf{o}}$$

Given $\mathbf{\Sigma}$, we have

$$
\begin{aligned}
&|\mathcal{L}(\mathbf{o}; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= |ln \frac{Pr[\mathcal{M}(\mathcal{D}, \mathcal{G}, \mathbf{\Sigma})] = \mathbf{o}}{Pr[\mathcal{M}(\mathcal{D}', \mathcal{G}, \mathbf{\Sigma})] = \mathbf{o}}| \\
&= |ln \frac{Pr[\mathcal{G}(\mathcal{D}) + \mathcal{N}(0, \mathbf{\Sigma})] = \mathbf{o}}{Pr[\mathcal{G}(\mathcal{D}') + \mathcal{N}(0, \mathbf{\Sigma})] = \mathbf{o}}| \\
&= |ln \frac{exp(-\frac{1}{2}(\mathbf{o} - \mathcal{G}(\mathcal{D}))^\intercal \mathbf{\Sigma}^{-1}(\mathbf{o} - \mathcal{G}(\mathcal{D}))}{exp(-\frac{1}{2}(\mathbf{o} - \mathcal{G}(\mathcal{D}'))^\intercal \mathbf{\Sigma}^{-1}(\mathbf{o} - \mathcal{G}(\mathcal{D}'))}|
\end{aligned}
$$

Given the transformation matrix $\mathbf{A}_{(i)}$, we denote $\mathbf{M}$ as $a^2 \mathbf{A}_{(i)} \mathbf{A}_{(i)}^\intercal$, which is a symmetric matrix. Therefore, $\mathbf{M}$ can be decomposed by Singular Value Decomposition as:

$$\mathbf{M} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\intercal,$$

where $\mathbf{U}, \mathbf{\Lambda} \in \mathbb{R}^{w \times w}$, and $\mathbf{\Lambda}$ is a diagonal matrix with each element on its diagonal being the singular values of $\mathbf{M}$. We construct a matrix $\mathbf{K} \in \mathbb{R}^{w \times w}$, s.t., $\mathbf{K} = \mathbf{U} \mathbf{\Lambda}^{\frac{1}{2}} \mathbf{U}^\intercal$, then

$$\mathbf{M} = \mathbf{K} \mathbf{K}^\intercal,$$

Let $\mathbf{p}$ denotes $\mathbf{o} - \mathcal{G}(\mathcal{D})$, then $\mathbf{p} \sim \mathcal{N}(0, \mathbf{\Sigma})$. Construct a Gaussian random variable $\tilde{\mathbf{p}} \sim \mathcal{N}(0, \sigma^2)$. We can have $\mathbf{p} = \mathbf{K} \tilde{\mathbf{p}}$. Given $\mathbf{v} = \mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')$, we have

$$
\begin{aligned}
&|\mathcal{L}(\mathbf{o}; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= \frac{1}{2} |(\mathbf{p} + \mathbf{v})^\intercal \mathbf{\Sigma}^{-1} (\mathbf{p} + \mathbf{v}) - \mathbf{p}^\intercal \mathbf{\Sigma}^{-1} \mathbf{p}|
\end{aligned}
$$

Construct $\hat{\mathbf{v}} = \mathbf{K}^{-1}\mathbf{v}$, then the above formula becomes

$$
\begin{aligned}
&|\mathcal{L}(\mathbf{o}; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= \frac{1}{2}|(\mathbf{K}\tilde{\mathbf{p}} + \mathbf{K}\hat{\mathbf{v}})^\intercal \mathbf{\Sigma}^{-1}(\mathbf{K}\tilde{\mathbf{p}} + \mathbf{K}\hat{\mathbf{v}}) - \mathbf{p}^\intercal \mathbf{\Sigma}^{-1}\mathbf{p}| \\
&= \frac{1}{2\sigma^2}|(\mathbf{K}\tilde{\mathbf{p}} + \mathbf{K}\hat{\mathbf{v}})^\intercal (\mathbf{K}\mathbf{K}^\intercal)^{-1}(\mathbf{K}\tilde{\mathbf{p}} + \mathbf{K}\hat{\mathbf{v}}) - \\
&\qquad (\mathbf{K}\tilde{\mathbf{p}})^\intercal (\mathbf{K}\mathbf{K}^\intercal)^{-1}(\mathbf{K}\tilde{\mathbf{p}})| \\
&= \frac{1}{2\sigma^2}|(\tilde{\mathbf{p}} + \hat{\mathbf{v}})^\intercal \mathbf{K}^\intercal (\mathbf{K}\mathbf{K}^\intercal)^{-1}\mathbf{K}(\tilde{\mathbf{p}} + \hat{\mathbf{v}}) - \\
&\qquad \tilde{\mathbf{p}}^\intercal \mathbf{K}^\intercal (\mathbf{K}\mathbf{K}^\intercal)^{-1}\mathbf{K}\tilde{\mathbf{p}}| \\
&= \frac{1}{2\sigma^2}|(\tilde{\mathbf{p}} + \hat{\mathbf{v}})^\intercal (\tilde{\mathbf{p}} + \hat{\mathbf{v}}) - \tilde{\mathbf{p}}^\intercal \tilde{\mathbf{p}}| \\
&= \frac{1}{2\sigma^2}|\|\tilde{\mathbf{p}}\|^2 - \|\tilde{\mathbf{p}} + \hat{\mathbf{v}}\|^2|,
\end{aligned}
$$

where $\tilde{\mathbf{p}}_k \sim \mathcal{N}(0, \sigma^2)$ and $\hat{\mathbf{v}} = \mathbf{K}^{-1}\mathbf{v}$. This is very similar to the proof in 6, except for the $\hat{\mathbf{v}}$. We can get the similar result as,

$$
\begin{aligned}
&|\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= \frac{1}{2\sigma^2}|\|\tilde{\mathbf{p}}\|^2 - \|\tilde{\mathbf{p}} + \hat{\mathbf{v}}\|^2| \\
&= \frac{1}{2\sigma^2}|\|\tilde{p'}_1 + \hat{\mathbf{v}}\|^2 - \|\tilde{p'}_1\|^2| \\
&= \frac{1}{2\sigma^2}|\|\tilde{p'}_1 + \mathbf{K}^{-1}\mathbf{v}\|^2 - \|\tilde{p'}_1\|^2| \\
&= \frac{1}{2\sigma^2}|\|\mathbf{K}^{-1}\mathbf{v}\|^2 + 2|\tilde{p'}_1\mathbf{K}^{-1}\mathbf{v}|| \\
&\leq \frac{1}{2\sigma^2}|\|\mathbf{K}^{-1}\|_2^2\Delta_{\mathcal{G}}^2 + 2|\tilde{p'}_1|\|\mathbf{K}^{-1}\|_2\Delta_{\mathcal{G}}| \\
&= \frac{1}{2\sigma^2}|\hat{\Delta}_{\mathcal{G}}^2 + 2|{p'}_1|\hat{\Delta}_{\mathcal{G}}|,
\end{aligned}
$$

where $\hat{\Delta}_{\mathcal{G}} = \|\mathbf{K}^{-1}\|_2\Delta_{\mathcal{G}} = S_{max}(\mathbf{K}^{-1})\Delta_{\mathcal{G}} = \frac{\Delta_{\mathcal{G}}}{S_{min}(\mathbf{M})^{\frac{1}{2}}}$, and $S_{min}(\mathbf{M})$ is the minimum singular value of $\mathbf{M}$

Following the proof of Theorem A.1 in [27], we can show that the mechanism $\mathcal{M}$ is

(ϵ, δ)-DP, with

$$\epsilon \in (0, 1], \sigma \geq \frac{\sqrt{2ln(1.25/\delta)}\hat{\Delta}_{\mathcal{G}}}{a\epsilon}.$$

Consequently, the theorem holds.

□

.6 Proof of Lemma 1 (Perturbation Transformation)

Proof. We leverage the Taylor expansion to rewrite $l(\mathbf{z}_{(i)}^{non}, \theta)$ at the data point $\mathbf{x}_{(i)}^{non}$ as follows,

$$\begin{aligned} &l(\mathbf{z}_{(i)}^{non}, \theta) = l(\mathbf{x}_{(i)}^{non}, \theta) + \\ &(\mathbf{z}_{(i)}^{non} - \mathbf{x}_{(i)}^{non})^{\intercal} \nabla_{\mathbf{x}_{(i)^{non}}} l(\mathbf{x}_{(i)}^{non}, \theta) + o(\mathbf{z}_{(i)}^{non} - \mathbf{x}_{(i)}^{non}) \end{aligned} \tag{3}$$

Since the only mild constraint on $l(\mathbf{z}_{(i)}^{non}, \theta)$ is C-Lipschitz continuous, the higher order terms $o(\mathbf{z}_{(i)} - \mathbf{x}_{(i)})$ is non-negative for "non-negative cases". Therefore, Equation 3 can be further approximated as,

$$l(\mathbf{z}_{(i)}^{non}, \theta) \geq l(\mathbf{x}_{(i)}^{non}, \theta) + (\mathbf{z}_{(i)}^{non} - \mathbf{x}_{(i)}^{non})^{\intercal} \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta), \tag{4}$$

which essentially tightens the privacy budget. Calculating the gradient of the above loss function, we can have:

$$\begin{aligned} &\nabla_{\theta} l(\mathbf{z}_{(i)}^{non}, \theta) \\ \geq &\nabla_{\theta} l(\mathbf{x}_{(i)}^{non}, \theta) + \nabla_{\theta}((\mathbf{z}_{(i)}^{non} - \mathbf{x}_{(i)}^{non})^{\intercal} \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta))x \\ = &\nabla_{\theta} l(\mathbf{x}_{(i)}^{non}, \theta) + (\mathbf{z}_{(i)}^{non} - \mathbf{x}_{(i)}^{non})^{\intercal} \mathbf{J}_{\theta} \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta), \end{aligned} \tag{5}$$

where $\mathbf{J}_{\theta} \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta)$ denotes the Jacobian matrix. We assume the gradient perturba-

tion $\mathbf{p}_{(i)}$ is defined as $\mathbf{p}_{(i)} = \mathbf{J}_\theta \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta)\mathbf{b}_{(i)}$, which is transformed from the input perturbation $\mathbf{b}_{(i)}$. The gradient can thus be rewritten as Equation 6.3:

$$\nabla_\theta l(\mathbf{z}_{(i)}^{non}, \theta) = \nabla_\theta l(\mathbf{x}_{(i)}^{non}, \theta) + \mathbf{p}_{(i)}.$$

To analyze the statistics of $\mathbf{J}_\theta \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta)$, we vectorize $\mathbf{b}_{(i)}$, $\mathbf{z}_{(i)}^{non}$ and θ, and let $\mathbf{A}_{(i)} = \mathbf{J}_\theta \nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta)$. Therefore, the problem becomes that given $\mathbf{p}_{(i)} = \mathbf{A}_{(i)}\mathbf{b}_{(i)}$, where $\mathbf{z}_{(i)}^{non}, \mathbf{b}_{(i)} \in \mathbb{R}^{v\times 1}$, $\mathbf{b}_{(i)}^{(k)} \sim \mathcal{N}(0, \sigma^2), \theta \in \mathbb{R}^{w\times 1}, \mathbf{A}_{(i)} \in \mathbb{R}^{v\times 1\times w}$, what is the scale of $\mathbf{p}_{(i)}$.

Denoting $\mathbf{M}_{(i)}$ as $\mathbf{A}_{(i)}\mathbf{A}_{(i)}^\intercal$, then according to the linear transformation of Gaussian random variable, we can conclude that $\mathbf{p}_{(i)} \sim \mathcal{N}(0, \mathbf{\Sigma}_{(i)})$, where $\mathbf{\Sigma}_{(i)}$ is the covariance matrix of $\mathbf{p}_{(i)}$ and $\mathbf{\Sigma}_{(i)} = \mathbf{M}_{(i)}\sigma^2$. □

.7 Theorem 7 (HGM) and Proof

Theorem 7. *Heterogeneous Gaussian Mechanism. Let $\mathcal{G} : \mathbb{R}^v \to \mathbb{R}^w$ be an arbitrary w-dimensional function, and $\Delta_\mathcal{G} = \max_{\mathcal{D},\mathcal{D}'} \|\mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')\|_2$. A Heterogeneous Gaussian Mechanism $\mathcal{M}$ with the diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{w\times w}$ adds noise to each of the w elements of the output. The mechanism $\mathcal{M}$ is (ϵ, δ)-DP, with*

$$\epsilon \in (0, 1], \sigma_{min} \geq \sqrt{2ln(1.25/\delta)}\Delta_\mathcal{G}/\epsilon.$$

where $\mathbf{\Sigma}$ is a diagonal matrix with each element being $\sigma_1^2, \sigma_2^2, ..., \sigma_w^2$ and $\sigma_{min} = \min_{i\in\{1,2,...,w\}} \sigma_i$.

The proof is as follows:

Proof. The privacy loss of an output o is defined as:

$$\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}') = ln\frac{Pr[\mathcal{M}(\mathcal{D}, \mathcal{G}, \Sigma)] = o}{Pr[\mathcal{M}(\mathcal{D}', \mathcal{G}, \Sigma)] = o}$$

Given Σ, we have

$$
\begin{aligned}
&|\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= |ln \frac{Pr[\mathcal{M}(\mathcal{D}, \mathcal{G}, \Sigma)] = o}{Pr[\mathcal{M}(\mathcal{D}', \mathcal{G}, \Sigma)] = o}| \\
&= |ln \frac{Pr[\mathcal{G}(\mathcal{D}) + \mathcal{N}(0, \Sigma)] = o}{Pr[\mathcal{G}(\mathcal{D}') + \mathcal{N}(0, \Sigma)] = o}| \\
&= |ln \frac{exp(-\frac{1}{2}(o - \mathcal{G}(\mathcal{D}))^\intercal \Sigma^{-1}(o - \mathcal{G}(\mathcal{D}))}{exp(-\frac{1}{2}(o - \mathcal{G}(\mathcal{D}'))^\intercal \Sigma^{-1}(o - \mathcal{G}(\mathcal{D}'))}|
\end{aligned}
$$

According to the theorem, Σ is a diagonal matrix with each element being $\sigma_1^2, \sigma_2^2, ..., \sigma_w^2$. We can construct a diagonal matrix $K \in \mathbb{R}^{w \times w}$, s.t., the diagonal elements of K are $\sigma_1, \sigma_2, ..., \sigma_w$, then

$$M = KK^\intercal,$$

$$K = K^\intercal.$$

Following the proof of Theorem 3, we can show that the mechanism $\mathcal{M}$ is (ϵ, δ)-DP, with

$$\epsilon \in (0, 1], 1 \geq \sqrt{2ln(1.25/\delta)} \hat{\Delta}_{\mathcal{G}} / \epsilon.$$

where $\hat{\Delta}_{\mathcal{G}} = \frac{\Delta_{\mathcal{G}}}{\sigma_{min}}$, and $\sigma_{min} = \min_{i \in \{1,2,...,w\}} \sigma_i$. Consequently, the theorem holds.

Next, we will prove that this mechanism indeed describe the same fact as HGM in [72] but from different views.

We define $\sigma^2 = \frac{1}{w} \sum_{i=1}^{w} \sigma_i^2$, and construct a vector $r \in \mathbb{R}^w$ with each element being $r_i = \frac{\sigma_i^2}{\sum_{i=1}^{w} \sigma_i^2}$. Then we have $\sigma_i^2 = \sigma^2 \times w r_i$, and each element of the diagonal matrix K becomes $\sqrt{w r_1}\sigma, \sqrt{w r_2}\sigma, ..., \sqrt{w r_w}\sigma$.

Given $v = \mathcal{G}(\mathcal{D}) - \mathcal{G}(\mathcal{D}')$ and v_i denotes i-th element of v, we construct a vector

$\hat{v} = \sqrt{\sum_{i=1}^{w} \frac{1}{wr_i} v_i^2}$. Then we can show that,

$$
\begin{aligned}
&|\mathcal{L}(o; \mathcal{M}, \mathcal{D}, \mathcal{D}')| \\
&= \frac{1}{2} | \|K^{-1}v\|^2 + 2|\tilde{p'}_1 K^{-1} v| | \\
&= \frac{1}{2\sigma^2} | \|\hat{v}\|^2 + 2|\tilde{p'}_1 \hat{v}| | \\
&\leq \frac{1}{2\sigma^2} | \hat{\Delta}_{\mathcal{G}}^2 + 2|p'_1| \hat{\Delta}_{\mathcal{G}} |,
\end{aligned}
$$

where $\hat{\Delta}_{\mathcal{G}} = \max_{\mathcal{D},\mathcal{D}'} \sqrt{\|\hat{v}\|^2}$ represents the new sensitivity.

Following the proof of Theorem 3, we can show the similar conclusion as in [72] that this mechanism is (ϵ, δ)-DP, with

$$
\epsilon \in (0, 1], \sigma \geq \sqrt{2ln(1.25/\delta)} \hat{\Delta}_{\mathcal{G}} / \epsilon.
$$

□

Compared with [72], Theorem 7 gives different definitions on the sensitivity $\Delta_{\mathcal{G}}$ and perturbation scale σ. However, we have shown in the proof that these two mechanisms indeed describe the same fact from different views.

.8 Proof of Lemma 2

In vanilla SGD, the algorithm picks one example at each iteration. Thus, the subscript t is eqivalent to (i). Lemma 2 can be easily derived from the following lemma:

Lemma 3. *Given perturbed example $\mathbf{z}_t^{non} = \mathbf{x}_t^{non} + \mathbf{b}_t$ with $\mathbf{b}_t^{(k)} \sim \mathcal{N}(0, \sigma^2)$, the number of training steps T, and C-Lipschitz continuous loss l. The gradient $\nabla_{\theta_t} l(\mathbf{z}_t^{non}, \theta_t)$ at each step of vanilla SGD can be reformulated as the gradient with respect to the original sample*

with a gradient perturbation:

$$\nabla_{\theta_t} l(\mathbf{z}_t^{non}, \theta_t) \geq \nabla_{\theta_t} l(\mathbf{x}_t^{non}, \theta_t) + \mathbf{p}_t, \tag{6}$$

where $\mathbf{p}_t$ *is the transformed perturbation with* $\mathbf{p}_t \sim \mathcal{N}(0, \mathbf{\Sigma}_t)$, $\mathbf{\Sigma}_t = T\mathbf{M}_t\sigma^2$, $\mathbf{M}_t = \mathbf{A}_{(i)}\mathbf{A}_{(i)}^\intercal$, $\mathbf{A}_{(i)} = \mathbf{J}_{\theta_t}\nabla_{\mathbf{x}_{(i)}^{non}} l(\mathbf{x}_{(i)}^{non}, \theta_t)$

The T in Lemma 3 comes from the fact that compared to traditional gradient perturbation added at each iteration, the input perturbation is only added at the start of training for one single time. The variance is amplified by a coefficient T when input perturbation is transformed into the gradient perturbation.

Given two neighbouring datasets $\mathcal{D}$ and $\mathcal{D}'$, training data size N, the transformation matrix $\mathbf{M}_t$ and minimum singular value $S_{min}(\mathbf{M}_t)$ are calculated to analyze DP contribution from input perturbation. However, because the sample at each iteration is randomly picked and $S_{min}(\mathbf{M}_t)$ is data-dependent, it is challenging to bound the difference of the transformed gradient perturbation p between the optimization processes on these two datasets.

To solve this, we first claim that the minimum singular value of the transformation matrix, $S_{min}(\mathbf{M}_t)$ determines how much DP guarantee MGM can provide. According to Corollary 1, this DP guarantee is equivalent to the guarantee provided by a traditional Gaussian Mechanism with its perturbation following a Gaussian distribution $\mathcal{N}(0, TS_{min}(\mathbf{M}_t)\sigma^2)$. Therefore, Lemma 2 is derived.

.9 Perturbation Transformation in Mini-batch SGD

In mini-batch SGD, the algorithm randomly picks a batch of samples at each iteration and feeds them into the model for optimization. Given the initial parameters θ_0, iteration t, the

parameters are updated as:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{B} \sum_{i=1}^{B} (\nabla_\theta l(\theta_t, \mathbf{z}_t^{non}) + \nabla_\theta l(\theta_t, \mathbf{z}_t^{neg})), \tag{7}$$

where η denotes the learning rate, B denotes the batch size, $\mathbf{z}_t^{non}$ denotes the perturbed samples of "non-negative cases" and $\mathbf{z}_t^{neg}$ denotes the perturbed samples of "negative cases". Both $\mathbf{z}_t^{non}$ and $\mathbf{z}_t^{neg}$ are randomly picked at iteration t.

According to the definition of $\nabla_{\theta_t} l(\theta_t, \mathbf{z}_t^{non})$ in Equation (6), we have that

$$\frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_t} l(\theta_t, \mathbf{z}_t^{non}) \geq \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta l(\mathbf{x}_t^{non}, \theta) + \mathbf{p}_t, \tag{8}$$

where $\mathbf{p}_t$ is the transformed perturbation with $\mathbf{p}_t \sim \mathcal{N}(0, \mathbf{\Sigma}_t)$, $\mathbf{\Sigma}_t = T\mathbf{M}_t\sigma^2$, $\mathbf{M}_t = \frac{1}{B^2} \sum_{i=1}^{B} \mathbf{A}_{(i)} \mathbf{A}_{(i)}^\intercal$, $\mathbf{A}_{(i)} = \mathbf{J}_{\mathbf{x}_{(i)}^{non}} \nabla_\theta l(\mathbf{x}_{(i)}^{non}, \theta)$.

This gradient is similar to that of vanilla SGD in Lemma 3, except that $\mathbf{M}_t$ is $\frac{1}{B^2} \sum_{i=1}^{B} \mathbf{A}_{(i)} \mathbf{A}_{(i)}^\intercal$ instead of $\mathbf{A}_{(i)} \mathbf{A}_{(i)}^\intercal$. We can directly follow the processes in vanilla SGD and derive Theorem 4 and Corollary 2 with $\mathbf{M}_t = \frac{1}{B^2} \sum_{i=1}^{B} \mathbf{A}_{(i)} \mathbf{A}_{(i)}^\intercal$.

According to Corollary 2, the DP guarantee that **MMGA** provides at iteration t for "non-negative cases" is equivalent to that provides by a **GM** with $\mathcal{N}(0, \xi_{up}^2)$.

On the other hand, we have the following for "negative cases":

$$\frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_t} l(\theta_t, \mathbf{z}_t^{neg}) = \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta l(\mathbf{x}_t^{neg}, \theta) + \mathcal{N}(0, \xi_{up}^2). \tag{9}$$

Therefore, we can also claim that Corollary 2 is applicable to both "non-negative cases" and "negative cases" in mini-batch SGD.

.10 Proof of Theorem 9 (MMGA with MA)

Given a multivariate Gaussian mechanism $\mathcal{M}$, two neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$, the output $\mathbf{o}$, we first define the log moments of the privacy loss random variable as follows,

$$m(\mathbf{o}; \mathcal{M}, \mathcal{D}, \mathcal{D}') \triangleq log \frac{Pr(\mathcal{M}(\mathcal{D}) \in \mathbf{o})}{Pr(\mathcal{M}(\mathcal{D}') \in \mathbf{o})}. \tag{10}$$

Then the log of the moment generating function at the value α is defined as follows,

$$\lambda_{\mathcal{M}}(\alpha; \mathcal{D}, \mathcal{D}') \triangleq log\, \mathbb{E}_{\mathbf{o} \sim \mathcal{M}(\mathcal{D})}(e^{\alpha m(\mathbf{o}; \mathcal{M}, \mathcal{D}, \mathcal{D}')}). \tag{11}$$

We take the maximum over all possible neighboring pairs of $\mathcal{D}$ and $\mathcal{D}'$ to obtain the moments accountant:

$$\lambda_{\mathcal{M}}(\alpha) \triangleq \max_{\mathcal{D}, \mathcal{D}'} \lambda_{\mathcal{M}}(\alpha; \mathcal{D}, \mathcal{D}'). \tag{12}$$

With the moments accountant $\lambda_{\mathcal{M}}(\alpha)$, we have the following lemma and theorem,

Theorem 8. *For any $\epsilon > 0$, the mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private for*

$$\delta = \min_{\alpha} exp(\lambda_{\mathcal{M}}(\alpha) - \alpha\epsilon).$$

The proof of Theorem 8 is the same as the proof of Theorem 2.2 in [2].

Lemma 4. *Given a function $f : \mathbb{R}^v \rightarrow \mathbb{R}^w$ with $\|f(.)\|_2 \geq 1$. Let $\xi_{up} \leq 1$, Ca be the gradient clipping coefficient and L be the sample from origin data with the sampling probability $q = \frac{B}{N} < \frac{1}{16\xi_{up}}$. Then for any positive integer $\alpha \leq \xi_{up}^2 ln \frac{1}{q\xi_{up}}$, the mechanism $\mathcal{M}(d) = \sum_{d_i \in L} f(d_i) + \mathcal{N}(0, \mathbf{\Sigma}_{(i)}\sigma^2)$ satisfies*

$$\lambda_{\mathcal{M}}(\alpha) \leq \frac{q^2\alpha(\alpha+1)}{(1-q)\xi_{up}^2} + \mathcal{O}(\frac{q^3\alpha^3}{\xi_{up}^3})$$

The detailed proof of the lemma 4 is in Appendix .11.

Theorem 9. *There exist constants c_1 and c_2 so that given the sampling probability $q = \frac{B}{N}$ and the number of training steps T, for any $\epsilon < c_1 q^2$, Algorithm 3 is (ϵ, δ)-differential private for any $\delta > 0$ if*

$$\xi_{up} \geq c_2 \frac{q\sqrt{Tlog(1/\delta)}}{\epsilon} \tag{13}$$

With Lemma 4, the proof is similar as the proof of Theorem 1 in [2].

.11 Proof of Lemma 4 (MA)

Proof. Given a function $f : \mathbb{R}^v \to \mathbb{R}^w$, we fix $\mathcal{D}$ and let $\mathcal{D} = \mathcal{D}' \cup d_{(n)}$. Without loss of generality, let $\|f(d_{(n)})\| = 1$ and $\sum_{i \in B \setminus [n]} f(d_{(i)}) = 0$, and let ζ_0 denotes the pdf of $\mathcal{N}(0, \Sigma)$, ζ_1 denotes the pdf of $\mathcal{N}(d_{(n)}, \Sigma)$, then we can have:

$$\begin{aligned} \mathcal{M}(\mathcal{D}') &\sim \zeta_0, \\ \mathcal{M}(\mathcal{D}) &\sim \zeta \triangleq (1-q)\zeta_0 + q\zeta_1. \end{aligned}$$

We want to show that

$$\mathbb{E}_{s \sim \zeta}[(\frac{\zeta(s)}{\zeta_0(s)})^\alpha] \leq \lambda, \tag{14}$$

$$\mathbb{E}_{s \sim \zeta_0}[(\frac{\zeta_0(s)}{\zeta(s)})^\alpha] \leq \lambda, \tag{15}$$

for certain λ.

For the Equation (14), we have

$$
\begin{aligned}
&\mathbb{E}_{s\sim\zeta}[(\frac{\zeta(s)}{\zeta_0(s)})^{\alpha}] \\
=&\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta(s)}{\zeta_0(s)})^{\alpha+1}] \\
=&\mathbb{E}_{s\sim\zeta_0}[1+(\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)})^{\alpha+1}] \\
=&\sum_{i=0}^{\alpha+1}\binom{\alpha+1}{i}\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)})^{i}].
\end{aligned} \tag{16}
$$

We find that the first term of (16) is 1, and the second term is 0 since

$$
\begin{aligned}
\mathbb{E}_{s\sim\zeta_0}[\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)}] &= \int_{\infty}^{-\infty}\zeta_0(s)\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)}\,\mathrm{d}s \\
&= \int_{\infty}^{-\infty}\zeta(s)-\zeta_0(s)\,\mathrm{d}s \\
&= \int_{\infty}^{-\infty}\zeta(s)\,\mathrm{d}s-\int_{\infty}^{-\infty}\zeta_0(s)\,\mathrm{d}s \\
&= 0
\end{aligned}
$$

The third term of (16) is:

$$
\begin{aligned}
&\binom{\alpha+1}{2}\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)})^{2}] \\
=&\binom{\alpha+1}{2}q^2\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)-\zeta_0(s)}{\zeta_0(s)})^{2}]
\end{aligned}
$$

Leveraging the fact that for any $a\in\mathbb{R}^v$, $\mathbb{E}_{s\sim\zeta_0}[exp(a^\intercal\Sigma^{-1}s)]=exp(a^\intercal\Sigma^{-1}a/2)$, then:

$$
\begin{aligned}
&\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)-\zeta_0(s)}{\zeta_0(s)})^{2}] \\
=&\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)}{\zeta_0(s)}-1)^{2}] \\
=&1-2\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)}{\zeta_0(s)})]+\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)}{\zeta_0(s)})^{2}]
\end{aligned} \tag{17}
$$

Given that $\zeta_0 \triangleq \mathcal{N}(0, \Sigma)$, $\zeta_1 \triangleq \mathcal{N}(d_{(n)}, \Sigma)$, we have

$$\begin{aligned}
\frac{\zeta_1(s)}{\zeta_0(s)} =& exp(-\frac{1}{2}((s-d_{(n)})^\intercal \Sigma^{-1}(s-d_{(n)}) - s^\intercal\Sigma^{-1}s)) \\
=& exp(-\frac{1}{2}(s^\intercal\Sigma^{-1}s - 2s^\intercal\Sigma^{-1}d_{(n)} + \\
& d_{(n)}^\intercal\Sigma^{-1}d_{(n)} - s^\intercal\Sigma^{-1}s)) \\
=& exp(-\frac{1}{2}(d_{(n)}^\intercal\Sigma^{-1}d_{(n)} - 2s^\intercal\Sigma^{-1}d_{(n)})) \\
=& exp(-\frac{1}{2}d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \times exp(s^\intercal\Sigma^{-1}d_{(n)})
\end{aligned}$$

Then (17) becomes:

$$\begin{aligned}
& \mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)-\zeta_0(s)}{\zeta_0(s)})^2] \\
=& 1 - 2\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)}{\zeta_0(s)})] + \mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)}{\zeta_0(s)})^2] \\
=& 1 - 2\mathbb{E}_{s\sim\zeta_0}[exp(-\frac{1}{2}d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \times exp(s^\intercal\Sigma^{-1}d_{(n)})] + \\
& \mathbb{E}_{s\sim\zeta_0}[exp(-d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \times exp(s^\intercal\Sigma^{-1}(2d_{(n)}))] \\
=& 1 - 2exp(-\frac{1}{2}d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \times exp(\frac{1}{2}d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) + \\
& exp(-d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \times exp(2d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \qquad (18) \\
=& 1 - 2 + exp(d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) \\
=& exp(d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) - 1 \qquad (19)
\end{aligned}$$

Therefore, the third term of (16) is:

$$\begin{aligned}
& \binom{\alpha+1}{2}\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)})^2] \\
=& \binom{\alpha+1}{2}q^2\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_1(s)-\zeta_0(s)}{\zeta_0(s)})^2] \\
=& \binom{\alpha+1}{2}q^2(exp(d_{(n)}^\intercal\Sigma^{-1}d_{(n)}) - 1) \qquad (20)
\end{aligned}$$

To bound (20), we need to determine that

$$\begin{aligned}
&\max_{d_{(n)}\in\mathbb{R}^v} d_{(n)}^{\intercal}\Sigma^{-1}d_{(n)} \\
=&\max_{d_{(n)}\in\mathbb{R}^v} d_{(n)}^{\intercal}\Sigma^{-\frac{1}{2}}\Sigma^{-\frac{1}{2}}d_{(n)} \\
=&\max_{d_{(n)}\in\mathbb{R}^v} (\frac{\|d_{(n)}^{\intercal}\Sigma^{-\frac{1}{2}}\|_2}{\|d_{(n)}\|_2})^2 \\
=&S_{max}(\Sigma^{-\frac{1}{2}})^2 \\
=&\frac{1}{S_{min}(\Sigma)},
\end{aligned} \tag{21}$$

Therefore, the third term of (16)

$$\begin{aligned}
&\binom{\alpha+1}{2}\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta(s)-\zeta_0(s)}{\zeta_0(s)})^2] \\
\leq&\binom{\alpha+1}{2}q^2(exp(\frac{1}{S_{min}(\Sigma)})-1) \\
\leq&\binom{\alpha+1}{2}q^2\frac{2}{S_{min}(\Sigma)} \\
=&\frac{q^2(\alpha+1)\alpha}{S_{min}(\Sigma)}
\end{aligned} \tag{22}$$

For the Equation (15), we have

$$\begin{aligned}
&\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_0(s)}{\zeta(s)})^\alpha] \\
=&\mathbb{E}_{s\sim\zeta}[(\frac{\zeta_0(s)}{\zeta(s)})^{\alpha+1}] \\
=&\mathbb{E}_{s\sim\zeta}[1+(\frac{\zeta_0(s)-\zeta(s)}{\zeta(s)})^{\alpha+1}] \\
=&\sum_{i=0}^{\alpha+1}\binom{\alpha+1}{i}\mathbb{E}_{s\sim\zeta}[(\frac{\zeta_0(s)-\zeta(s)}{\zeta(s)})^i].
\end{aligned} \tag{23}$$

Similar to the analysis of (14), the first term of (23) is 1, and the second term is 0. The

third term is slightly different:

$$\begin{aligned}&\binom{\alpha+1}{2}\mathbb{E}_{s\sim\zeta}[(\frac{\zeta_0(s)-\zeta(s)}{\zeta(s)})^2]\\=&\binom{\alpha+1}{2}q^2\mathbb{E}_{s\sim\zeta}[(\frac{\zeta_0(s)-\zeta_1(s)}{\zeta(s)})^2]\end{aligned}$$

Because $\zeta \triangleq (1-q)\zeta_0 + q\zeta_1$, we can get that $\zeta(s) \geq (1-q)\zeta_0(s)$, and rewrite the above equation as:

$$\begin{aligned}&\binom{\alpha+1}{2}\mathbb{E}_{s\sim\zeta}[(\frac{\zeta_0(s)-\zeta(s)}{\zeta(s)})^2]\\\leq&\binom{\alpha+1}{2}\frac{q^2}{1-q}\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_0(s)-\zeta_1(s)}{\zeta_0(s)})^2]\end{aligned}$$

The following analysis is similar to that of (14), and we obtain the third term of (23)

$$\begin{aligned}&\binom{\alpha+1}{2}\mathbb{E}_{s\sim\zeta}[(\frac{\zeta_0(s)-\zeta(s)}{\zeta(s)})^2] &(24)\\\leq&\binom{\alpha+1}{2}\frac{q^2}{1-q}\mathbb{E}_{s\sim\zeta_0}[(\frac{\zeta_0(s)-\zeta_1(s)}{\zeta_0(s)})^2]\\\leq&\frac{q^2(\alpha+1)\alpha}{(1-q)S_{min}(\Sigma)} &(25)\end{aligned}$$

Compared with the proof of Lemma 3 in [2], it's obvious that the only difference between Gaussian mechanism and multivariate Gaussian mechanism is the noise scale. The covariance Σ of MGM can be decomposed into singular vectors and singular values, and the minimum singular value $S_{min}(\Sigma)$ actually solely determines the upper bound of the moment accountant $\lambda_{\mathcal{M}}(\alpha)$, which is the counterpart of $C\sigma^2$ of GM with C being the gradient clipping norm.

Following the proof of Lemma 3 in [2], and given $\Sigma \triangleq \frac{S_{min}(M)\sigma^2}{C^2}$, we can get Lemma 4:

Given a function $f : \mathbb{R}^v \to \mathbb{R}^w$ with $\|f(.)\|_2 \geq 1$. Let $\frac{S_{min}(M)\sigma^2}{C^2} \leq 1$, C be the gradient clipping norm and L be the sample from origin data with the sampling probability $q <$

$\frac{1}{16\sqrt{\frac{S_{min}(M)\sigma^2}{C^2}}}$. Then for any positive integer $\alpha \leq \frac{S_{min}(M)\sigma^2}{C^2} ln \frac{1}{q\sqrt{\frac{S_{min}(M)\sigma^2}{C^2}}}$, the mechanism $\mathcal{M}(d) = \sum_{d_i \in L} f(d_i) + \mathcal{N}(0, \Sigma\sigma^2)$ satisfies

$$\lambda_{\mathcal{M}}(\alpha) \leq \frac{q^2\alpha(\alpha+1)}{(1-q)\frac{S_{min}(M)\sigma^2}{C^2}} + \mathcal{O}(\frac{q^3\alpha^3C^3}{S_{min}(M)^{\frac{3}{2}}\sigma^3})$$

□

.12 The implementation of C-Lipschitz

In practice, the gradient $\nabla_\theta l(\mathbf{z}_{(i)}, \theta)$ is always clipped by a norm threshold C:

$$Clipped(\nabla_\theta l(\mathbf{z}_{(i)}, \theta)) \triangleq \frac{\nabla_\theta l(\mathbf{z}_{(i)}, \theta)}{\max(1, \|\nabla_\theta l(\mathbf{z}_{(i)}, \theta)\|_2 / C)} \tag{26}$$

Lemma 5. *Given perturbed example $\mathbf{z}_{(i)} = \mathbf{x}_{(i)} + \mathbf{b}_{(i)}$ with $\mathbf{b}_{(i)}^{(k)} \sim \mathcal{N}(0, \sigma^2)$, and denoting a clipping coefficient $a = 1/\max(1, \|\nabla_\theta l(\mathbf{z}_{(i)}, \theta)\|_2 / C)$. The gradient $\nabla_\theta l(\mathbf{z}_{(i)}, \theta)$ clipped by a norm threshold C can be reformulated as the gradient with respect to the original sample with a gradient perturbation:*

$$Clipped(\nabla_\theta l(\mathbf{z}_{(i)}, \theta)) = a\nabla_\theta l(\mathbf{x}_{(i)}, \theta) + \mathbf{p}_{(i)}, \tag{27}$$

where $\mathbf{p}_{(i)}$ is the transformed perturbation with $\mathbf{p}_{(i)} \sim \mathcal{N}(0, \mathbf{\Sigma}_{(i)})$, $\mathbf{\Sigma}_{(i)} = \mathbf{M}_{(i)}\sigma^2$, $\mathbf{M}_{(i)} = a^2\mathbf{A}_{(i)}\mathbf{A}_{(i)}^\intercal$ and $\mathbf{A}_{(i)} = \nabla_{\mathbf{x}_{(i)}} \nabla_\theta l(\mathbf{x}_{(i)}, \theta)$.

.13 Parameters Slicing

The AE can have a complex structure in order to remove the perturbation and reconstruct the input samples effectively. This complex structure results in a high-dimensional parameter space, and leads to an inefficient perturbation transformation. According to the

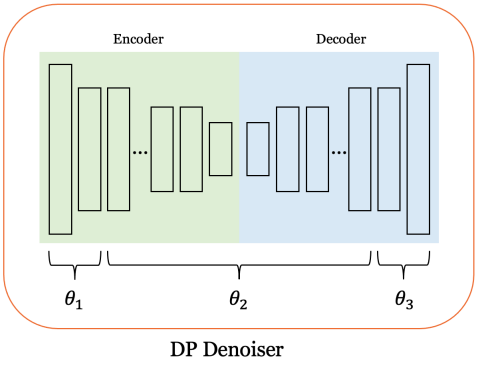


Figure 7: Denoiser with sliced parameters

definition of $A_{(i)}$ and M, when the dimension of θ is high, the calculation will incur a significant computation cost.

A solution we employ is to slice the parameter $\theta \in \mathbb{R}^w$ into several parts, e.g., $\theta_1 \in \mathbb{R}^{w_1}$, $\theta_2 \in \mathbb{R}^{w_2}$ and $\theta_3 \in \mathbb{R}^{w_3}$, corresponding to the first, intermediate, and last layers, respectively, as shown in Figure 7. The input perturbation transformation is then applied on θ_1, θ_2 and θ_3 separately with each sub-parameter slice having smaller size.

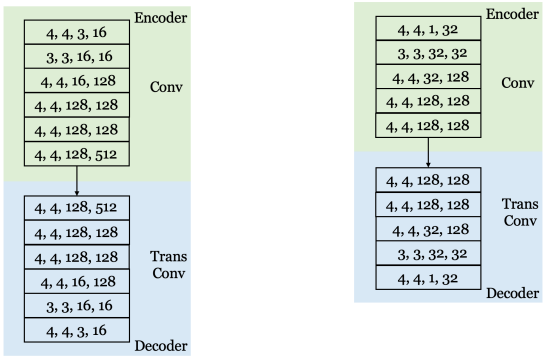
In the training stage, different optimizers work on different parameter slices and update them independently. Because the optimization is separated and each one can be regarded as an independent process, the perturbation transformation and MGM in Section 6.2.2 and MMGA in Section 6.2.3 can be applied to each process respectively. Without loss of generality, we use θ directly in the following subsections and sections.

.14 Detailed Experiments of Work 3

Pre-trained classifiers. Pre-trained classifiers are trained on public datasets. The pre-trained classifier for MNIST is a customized 12 layers deep convolutional network with residual blocks, which achieves 99.3% accuracy on test dataset. The pre-trained classifier for CIFAR-10 is a deep convolutional network transfered from VGG16 [78]. We replace the top layers of VGG16 with customized fully connected layers and initialize all bottom

layers with VGG16 parameters. This classifier achieves 86% accuracy on test dataset. Both classifiers are treated as public in all experiments.

Denoiser. We use convolutional and transposed convolutional layers to build the autoencoder based denoiser for both MNIST and CIFAR10 datasets. Layer normalization [7], residual [40] and skip structures are applied to avoid gradient vanishment. Drop out technique is applied to mitigate overfitting. The details of these two denoisers are shown in Figure 8a and Figure 8b.



(a) Detailed architecture of denoiser for CIFAR10

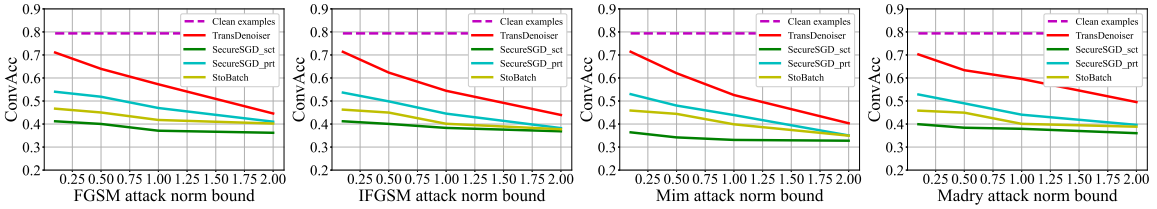
(b) Detailed architecture of denoiser for MNIST

Figure 8: Denoiser for two datasets, where each box denotes the layer, the digits in each box denotes the shape of convolutional filter, "Conv" denotes and "Trans Conv" denotes the boxes are convolutional and transposed convolutional layers respectively.

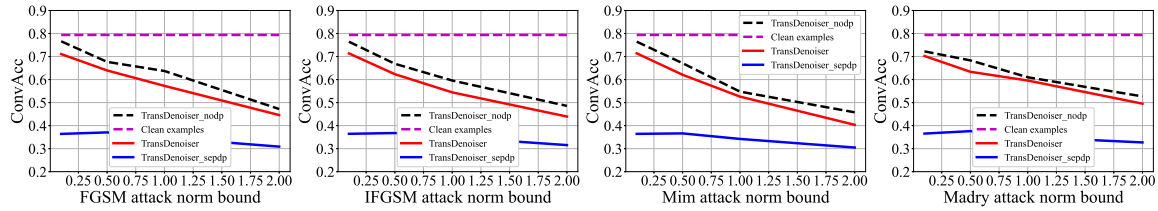
Implementation details. All models are implemented using Tensorflow 1.14 and trained with a system equipped with Nvidia V-100 GPU. We use the open source code of SecureSGD (`https://github.com/haiphanNJIT/SecureSGD`) and StoBatch (`https://github.com/haiphanNJIT/StoBatch`) to conduct experiments, and we fix an error in SecureSGD code where variance is used incorrectly as standard deviation. Our code is available at `https://anonymous.4open.science/r/14d3ec7c-ab5b-4c76-be92-1f3b0454563a/`.

Empirical defense. We show ConvAcc against four different adversarial attacks: FGSM, I-FGSM, Momentum Iterative Method (MIM) and MadryEtAl. As can be seen in Figure 9, the results for MIM and MadryEtAl are similar to those shown in Section 6.3 (Figure 6.3).

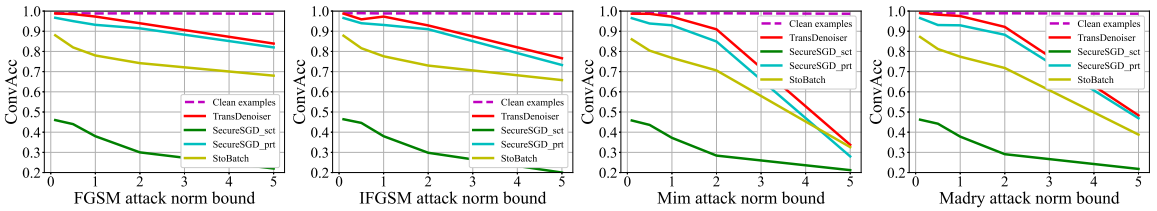
dominates all baselines and ablation cases over four different attacks and different attack norm bound.



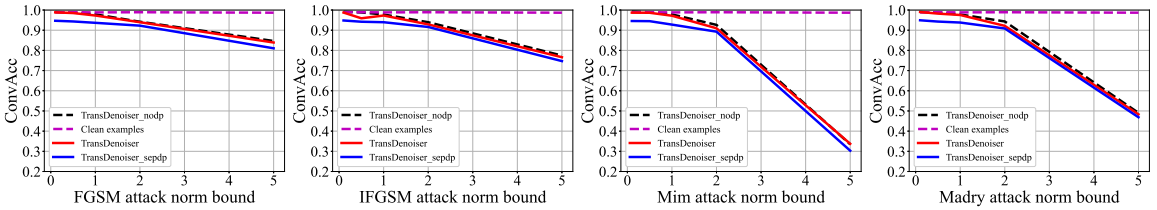
(a) and baselines on CIFAR10



(b) and ablation cases on CIFAR10



(c) and baselines on MNIST



(d) and ablation cases on MNIST

Figure 9: More comparison among , baselines and ablation cases for conventional accuracy vs. l_2 radii on two datasets. The input perturbation scale on CIFAR10 = 0.1, on MNIST = 0.25, the overall gradient perturbation scale = 2.0 ($\geq$ 2.0 for), and guarantee $(1.0, 1\mathrm{e}{-5})$-DP for private models.