Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Differentially private GANs by adding noise to Discriminator's loss

Chunling Han[a,b,*], *Rui Xue*[a]

[a] *SKLOIS, Institute of Information Engineering, CAS; School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China*

[b] *Indiana University Bloomington, IN 47401, USA*

ARTICLE INFO

Article history:
Received 24 October 2020
Revised 22 March 2021
Accepted 7 May 2021
Available online 18 May 2021

Keywords:
Generative Adversary Networks
Differential privacy
Discriminator's loss
Generator
Efficiency

ABSTRACT

Differentially private generative adversary network (GAN) is a very promising field in data privacy, with many practical real-world applications. The idea of differentially private GAN is to provide differential privacy protection for sensitive training datasets. By using differentially private GANs, training datasets can be protected from being remembered or encoded into GAN's parameters or generated data. Therefore, generated data can be safely used for data augmentation or replacing real sensitive data with very little privacy loss. However, existing methods for differentially private GANs are notoriously inefficient, most of them use a modified Tensorflow library, i.e., Tensorflow Privacy provided by Google, which is not only hard for programming but also very time-consuming even on TPU. In this paper, we provide a simpler and more efficient way to achieve differentially private GANs: when we set the training process of a GAN under a certain manner, we can use discriminator's loss as vehicle to achieve differential privacy. Compared with existing methods, our method needs no modification on Tensorflow core functions and will not significantly drag down the training process. We test our method on GANs for real-world datasets MNIST, Fashion MNIST, and SVHN datasets, experimental results show that our method is easy to implement, more practical, and more efficient than existing methods. With these advantages, we believe our method for differentially private GANs will be widely used in the very near feature.

1. Introduction

Generative Adversary network (GAN) is a thriving field with many practical applications. It can generate synthetic fake data to mimic real training data, which can be used for data augmentation or replacing real sensitive data to be released for outside research. For example, a world organization would like to analyze covid-19 patents' records for better understanding of this disease and making further predictions. Due to concerns of patients' privacy, medical institutes may not be willing to disclose their patients' records. In this case, by using generators of GANs, medical institutes can generate synthetic fake data to provide and contribute to the world organization.

However, generators of GANs might steal sensitive information from training datasets or reflect sensitive features into generated data. For example, generators may simply repeat the real sensitive data or encode features of real sensitive data into parameters. To address this problem, differentially private GANs become a promising solution. By applying differential privacy in GANs, sensitive training datasets can be protected, with little privacy loss, from being remembered or reflected into generators or generated data.

* Corresponding author.
E-mail addresses: hanchunling@iie.ac.cn (C. Han), xuerui@iie.ac.cn (R. Xue).
https://doi.org/10.1016/j.cose.2021.102322

Why existing works fail. From previous literature reviews, we find that most of existing methods for differentially private GANs are inefficient and will significantly drag down the training process. Those methods either add Gaussian noise to gradients during training process (Augenstein et al., 2019; Beaulieu-Jones et al., 2019; Frigerio et al., 2019; Torkzadehmahani et al., 2019; Xie et al., 2018; Xu et al., 2019; Zhang et al., 2018) (method named as DP-SGD (Abadi et al., 2016)) or train distributed teacher models to transfer knowledge to generators (Jordon et al., 2018; Long et al., 2020) (method named as PATE (Papernot et al., 2016; 2018)).

For DP-SGD category, dp-GAN (Zhang et al., 2018) uses DP-SGD method directly, clips the gradients and add noise. They also made some optimization to improve the training stability and convergence rate, including: adaptive clipping bounds adjusted to parameters during training by using a small set of auxiliary public data; parameter grouping to cluster parameters with similar clipping bounds, use a uniform clipping bound for each cluster to achieve a trade-off between privacy loss and convergence; warm starting to initialize the model with a small set of public data to accelerate the convergence. DP-GAN (Xie et al., 2018) uses a modified DP-SGD method, they applying differential privacy by clipping weights w, although this method does not clip the gradients directly as in DP-SGD, the authors show that by clipping w to a bound, the gradients are automatically bounded by some constant. GAN-obfuscator (Xu et al., 2019) develops the differentially private GAN on the improved WGAN and adopts adaptive clipping as in dp-GAN. dp-GAN-TSCD (Frigerio et al., 2019) use DP-SGD algorithm to design differentially private GAN to generate "time series, continuous, and discrete" data, such as UCI Adult and Mushroom data. DP-CGAN (Torkzadehmahani et al., 2019) develops a differentially private Conditional GAN (DP-CGAN) to generate labelled data. While SPRINT-gan (Beaulieu-Jones et al., 2019) develops differentially private AC-GAN (Auxiliary Classifier GAN) to generate labelled data. DP-FedAvg-GAN (Augenstein et al., 2019) uses DP-SGD algorithm for federated learning GANs.

For differentially private GAN using PATE, such as PATE-GAN (Jordon et al., 2018), it adopts the PATE method to achieve differential privacy, train distributed discriminators to transfer knowledge to the generator. PATE-GAN uses a new way to train the discriminator, it does not need auxiliary public dataset. Teacher discriminators are trained normally on disjoint partitions of training data, they only have access to their part of data. A student discriminator is trained with generated samples, labeled by the teachers using the PATE method. The generator is trained to minimize its loss with respect to the student discriminator. As a result, the student model can be trained privately without public data and the generator can utilize the process to improve the generated samples.

In terms of performance and efficiency, when using DP-SGD to achieve differentially private GANs, implementation needs to use a modified Tensorflow library, i.e., Tensorflow Privacy[1], which is known as very time-consuming even on TPU. When using multiple teacher models to transfer knowledge (PATE), it needs to train multiple GAN structures, involve in majority voting, which are also inefficient. These obstacles motivate our work.

From state-of-the-art works for differentially private GANs, we find that because the discriminator in a GAN has access to sensitive training datasets, most existing works for differentially private GANs focus on the discriminator, apply differential privacy on discriminator's parameters. Since the discriminator is differentially private, through propagation, the generator will also be differentially private.

However, as we know, after training a GAN, we only use the generator to generate fake data. Whether the discriminator is differentially private or not is actually not the key for a differentially private GAN. In fact, the real purpose of a differentially private GAN is actually to achieve a differentially private generator, then use the generator to generate differentially private fake data.

Therefore, in this paper, **we start from a new perspective and put our focus on the generator. Our strategy for a differentially private GAN is to achieve a differentially private generator.** We use differentially private generator to indicate differentially private GAN for the rest of this paper. Notice that, if the discriminator is also requested to be differentially private, our method will not fulfill that requirement. The overview of our approach is depicted in Fig. 1.

As shown in Fig. 1, the prerequisite of our approach is that we let the training process of GANs be as following steps:

- Generator generates a batch of fake data, discriminator is trained on this batch of fake data and a batch of real data, then set the discriminator untrainable.
- Generator generates a new batch of fake data, discriminator computes loss on this batch of fake data.
- GAN computes gradients according to the loss value, updates the generator.

Notice that, this training process is not a tailored process we made for our method. Actually, it is a very common training process that many GANs are using in real world now. **Under this training process, generator's gradients are derived from discriminator's loss on the second batch of generated data, if we set this discriminator's loss differentially private, through propagation, the generator's parameters would also be differentially private. Therefore, we can obtain a differentially private generator, and use this generator to generate differentially private fake data.**

To use discriminator's loss on the second batch of generated data as vehicle to apply differential privacy, we firstly clip discriminator's loss, then add Gaussian noise to the clipped discriminator's loss.

By using discriminator's loss as target, unlike most existing methods, we do not need to modify model optimizer functions in Tensorflow to change the way of updating gradients, also the computation workload for discriminator's loss is much less than the computation workload for numerous gradients, which brings our method high efficiency.

We evaluate our method on GANs for real-world datasets MNIST and Fashion MNIST, test different Gaussian noise levels to achieve different privacy losses. We use the same privacy loss measurement i.e. **Moment Accountant (MA)** (Abadi et al., 2016) used in previous proposed differentially private GANs

[1] https://github.com/tensorflow/privacy.

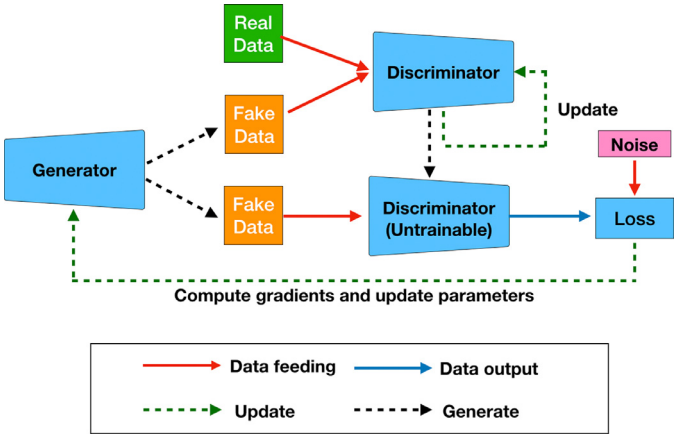

Fig. 1 – The overview of our approach to achieving differentially private GANs.

(DP-GAN (Xie et al., 2018), dp-GAN (Zhang et al., 2018)) to measure privacy loss of our scheme. Experimental results show that, with low privacy loss, our differentially private GANs can generate high-quality generated data. For example, with privacy loss at 6.786, generated data can achieve deep learning models with 98.03%, 85.24% and 88.65% accuracies for MNIST, Fashion MNIST and SVHN respectively. Compared with baselines for MNIST 99.20%, which is deep learning models trained on real training dataset, generated data from our differentially private GAN achieves close accuracy. While the performance of generated data for Fashion MNIST and SVHN dataset which achieves accuracies as 85.24% and 88.65%, are a little bit inferior to the baselines (92.40%) of Fashion MNIST and (96.74%) of SVHN.

Compared with existing methods, our method achieves prominent accuracies and brings high efficiency. Some existing methods need hours to train a differentially private GAN on MNIST, **while our method needs only a few minutes.**

The contributions of our work are as follows:

- Under a prerequisite training process, we propose a new and efficient method to achieve differentially private GANs.
- We use discriminator's loss as vehicle to apply differential privacy. Therefore, there is no need to modify Tensorflow Library and will not significantly drag down the training process.
- We evaluate the performance of our scheme on real-world datasets, experiments show that our method can effectively achieve high-quality differentially private GANs with low privacy loss.

The organization of the remainder of this paper is structured as follows: Preliminaries are introduced in Section 2. Our scheme along with security proof and privacy loss measurement are presented in Section 3. Then followed by Evaluations in Section 4. Comparison with related works is presented in Section 5. Finally, Conclusion is given in Section 6.

2. Preliminaries

In this section, we briefly describe the building blocks of our approach, including differential privacy and generative adversary network (GAN).

2.1. *Differential privacy*

Differential privacy (Dwork and Lei, 2009; Dwork and Rothblum, 2016; Dwork et al., 2010) is a standard for randomized algorithms analyzing dataset providing privacy guarantees. We recall the definition defined in Dwork et al. (2006).

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

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta \tag{1}$$

This definition is based on (ε, δ)-differential privacy, which means the plain ε-differential privacy can be broken with a probability δ. Two adjacent datasets means they only differ in a single entry.

To achieve differential privacy for a function, a common method is to add random noise to the function, the magnitude of the noise should be calibrated to the sensitivity of the function.

Definition 2. For $f : \mathcal{D} \to \mathcal{R}$, adjacent $d, d' \in \mathcal{D}$, the L_2 sensitivity of f is

$$\Delta_2 f = \max_{d,d \in \mathcal{D}} \| f(d) - f(d') \|_2 \tag{2}$$

where $||||_2$ means l_2 -norm

The Gaussian noise mechanism achieving differential privacy is defined as follows:

$$\mathcal{M}_f(D) \triangleq f(D) + \mathcal{N}(0, \Delta_2 f^2 \sigma^2) \tag{3}$$

where $\mathcal{N}(0, \Delta_2 f^2\sigma^2)$ is the normal (Gaussian) distribution with mean 0 and standard deviation $\Delta_2 f\sigma$.

2.2. GAN

Generative Adversarial Network (GAN) (Goodfellow et al., 2014) is a newly (2014) invented architecture in machine learning, used for training generative models. The idea of GAN is to let two neural networks compete with each other in a game, one learns to generate fake data while the other one learns to distinguish between real and fake data. Given a training set, the outcome of a GAN is a generative network that can generate new data with the same statistics as the training set.

Generative Adversarial Networks (GAN) consists of two models: a generator G and a discriminator D. Generator G takes random noise $z \sim p_z(z)$ as input, tries to output fake samples of data with distribution approximates real data's distribution $x \sim p_{data}(x)$. The discriminator D will estimate the probability that a sample is a real data comes from the training dataset rather than a fake data generated from G. These two models are simultaneously trained in a competitive way, the goal of a GAN is to train G and D playing a two-player minmax game with the value function V(G,D):

$$\min_G \max_D V(G,D) = E_{x\sim p_{\text{data}}(x)}[\log(D(x))] + E_{z\sim p_z(z)}[1-\log(D(G(z)))] \quad (4)$$

There might be differences on how to train GANs, in our paper, we use the following steps to train a GAN. In every training epoch:

1. Discriminator is trained on a batch of real data and a batch of fake data. Real data comes from training dataset and fake data comes from the Generator.
2. Set the Discriminator untrainable (parameters will not be updated).
3. The Generator and the Discriminator are combined as a GAN. Generator generates a new batch of fake data and feeds it to the Discriminator.
4. The GAN computes loss on that batch of fake data and computes gradients.
5. The Generator updates its parameters using the gradients.

3. Our approach

In this section, we illustrate our approach for differentially private GANs, then provide a security proof for our scheme and the measurement for privacy loss.

The goal of differentially private GAN is to protect sensitive training datasets, prevent generators and generated samples from stealing or reflecting sensitive features of training data. Our differentially private GANs should meet this requirement.

3.1. *Our approach to differentially private GANs*

Our method for differentially private GANs is described in Algorithm 1. The perspective of Algorithm 1 is that even though the generator has no access to training dataset, it has access to discriminator's parameters, which might have encoded sensitive features of training data. To prevent the generator from stealing sensitive information from discriminator's parameters, reflecting them to generated data, we need to make sure the derivatives from discriminator's parameters to generator's parameters contain as little as possible sensitive features. In Algorithm 1, we provide a differentially private parameter update method for the generator.

Algorithm 1 Differentially Private GAN

Input: Discriminator $D(\theta)$, generator $G(\phi)$, real dataset $\{\text{real\_data}_i\}_T$, loss function $D(\theta)$, loss norm bound C, Gaussian noise level σ.

Parameter: Initialize θ and ϕ.

1: **for** epoch t in range $(0, T)$ **do**
2: **Generator generates a batch of fake data**
3: $\text{fake\_data}_t \leftarrow G(z), z \sim p(z)$
4: **Train discriminator D** on a batch of real data $\textbf{real\_data}_t$ and the batch of fake data $\textbf{fake\_data}_t$, batch sizes are S.
5: Discriminator D calculates the loss: $d\_\text{real}_{t_i} \leftarrow D(\text{real\_data}_{t_i})$ and $d\_\text{fake}_{t_i} \leftarrow D(\text{fake\_data}_{t_i})$
6: $grad_{t_i} = \frac{1}{2}(grad\_\text{real}_{t_i} + grad\_\text{fake}_{t_i})$.
7: Set the discriminator untrainable.
8: The generator generates a batch of fake data from noise z. $\text{fake}'_t \leftarrow G(z), z \sim p(z)$, with size of S.
9: **Compute loss**
10: The discriminator predicts on the generated fake data and calculates the loss.
11: $g\_loss_i \leftarrow D(\text{fake}'_{t_i})$
12: **Clip loss**
13: $g\_loss_i \leftarrow g\_loss_i / \max(1, \frac{\|g\_loss_i\|_2}{C})$
14: **Add noise**
15: $g\_loss \leftarrow \frac{1}{S}(\sum g\_loss_i + \mathcal{N}(0, C^2\sigma^2))$
16: **Compute gradients**
17: The discriminator D and the generator G combine as a GAN, then GAN calculates the gradients according to g_loss.
18: **Generator updates parameters**
19: The generator updates its parameters according to the gradients.
20: **end for**

Output: Differentially private generator, compute the overall privacy loss (ε, δ) using moments accountant [1].

As shown in Algorithm 1, we need the following steps in every epoch of GAN training:

- Train the discriminator on a batch of fake data from the generator and a batch of real data from the training dataset, compute loss and update the discriminator, then set the discriminator untrainable.
- Generator G generates a batch of fake data fake'_t, the discriminator D predicts on the fake data, computes the loss $g\_loss \leftarrow D(\text{fake}'_t)$.
- Clip loss norm to a threshold C: if $\|g\_loss_i\|_2 \le C$ then g_loss_i would be preserved; if $\|g\_loss_i\|_2 \ge C$, then g_loss_i would be clipped to be norm of C.

Sum up the loss g_loss_i and then add Gaussian noise (with mean 0 and standard variance σ) to the sum of the clipped loss.

- Use the loss to compute gradients, generator G updates its parameters according to the gradients.

In Algorithm 1, we use Gaussian differential privacy mechanism to achieve a differentially private generator. Because the generator is differentially private, through propagation, the generated samples from the generator will also be differentially private.

3.2. *Soundness and security proof*

In this subsection, we explain the reason why under the training process prerequisite, applying differential privacy to discriminator's loss can be sufficient to preserve a differentially private generator.

In contrast, most of existing methods for differentially private deep learning models, GANs are also included, choose **gradients** to apply differential privacy, and use DP-SGD (differentially private stochastic gradient descent) to update parameters.

Notice that, if we change DP-SGD algorithm and apply differential privacy to loss functions instead of gradients, we would not achieve a differentially private deep learning model. The reason is because the gradients are computed by backpropagation. Gradients at each layer depends on activation values of the previous layer. These activation values are computed in the forward pass and the first layer activation value is computed on the **input data without noise**. To simply explain, we use Z_1 to indicate the first layer activation value, W_1 are the first layer parameters, X are input data, σ is the activation function (not the same σ used in Algorithm 1), A_1 is an intermediate value, that $A_1 = W_1X$. Then $Z_1 = \sigma(A_1)$. Through backpropagation, gradient for parameter W_1, denoted as ∂W_1, will have X as coefficients. In this case, X are input training data without noise, which makes the intent of providing differential privacy for training data X fail. Since the gradient for W_1 is not differentially private, the parameters of deep learning model will not be differentially private either, therefore we can not achieve a differentially private deep learning model.

Now let's see why applying differential privacy to **loss** in our method can guarantee a differentially private generator. The key is we use the prerequisite training process for GANs, we list the prerequisite training process here again.

1. Discriminator is trained on a batch of real data and a batch of fake data. Real data comes from training dataset and fake data comes from the Generator.
2. Set the Discriminator untrainable (parameters will not be updated).
3. The Generator and the Discriminator are combined as a GAN. Generator generates a new batch of fake data and feeds it to the Discriminator.
4. The GAN computes loss on that batch of fake data and computes gradients.
5. The Generator updates its parameters using the gradients.

Under this training process, we explain why differentially private discriminator's loss on the second batch of generated data can sufficiently guarantee a differentially private generator.

As we know, the discriminator is trained based on training datasets, sensitive information of training data could be encoded or reflected into discriminator's parameters. While, the gradients for generator's parameters are computed exactly based on discriminator's parameters. Through this connection, the generator could steal some sensitive information from the discriminator.

Now, we use a simple example shown in Fig. 2 to demonstrate how differentially private discriminator's loss on the second batch of generated data (denoted as E in Fig. 2) will affect generator's gradients. Notice that, this is an example for demonstration, not a real structure used in our GAN structure.

We take generator's parameter W_{11} (shown in Fig. 2) as an example to compute gradient for W_{11}. The chain rule will be:

$$\begin{aligned}\frac{\partial E}{\partial W_{11}} &= \frac{\partial E}{\partial A'_{31}}\frac{\partial A'_{31}}{\partial Z'_{31}}\frac{\partial Z'_{31}}{\partial W'_{31}}\frac{\partial W'_{31}}{\partial A'_{21}}\frac{\partial A'_{21}}{\partial Z'_{21}}\cdots\frac{\partial Z_{21}}{\partial W_{21}}\frac{\partial W_{21}}{\partial A_{11}}\frac{\partial A_{11}}{\partial Z_{11}}\frac{\partial Z_{11}}{\partial W_{11}}\\ &+\frac{\partial E}{\partial A'_{31}}\frac{\partial A'_{31}}{\partial Z'_{31}}\frac{\partial Z'_{31}}{\partial W'_{31}}\frac{\partial W'_{31}}{\partial A'_{22}}\frac{\partial A'_{22}}{\partial Z'_{22}}\cdots\frac{\partial Z_{22}}{\partial W_{21}}\frac{\partial W_{21}}{\partial A_{11}}\frac{\partial A_{11}}{\partial Z_{11}}\frac{\partial Z_{11}}{\partial W_{11}}\\ &+\ldots\\ &\ldots\end{aligned} \tag{5}$$

We take one specific line to explain how the differentially private loss will affect gradients (shown as the green line in Fig. 2). As we can see, the chain rule is:

$$\frac{\partial E}{\partial A'_{31}}\frac{\partial A'_{31}}{\partial Z'_{31}}\frac{\partial Z'_{31}}{\partial W'_{31}}\frac{\partial W'_{31}}{\partial A'_{21}}\frac{\partial A'_{21}}{\partial Z'_{21}}\frac{\partial Z'_{21}}{\partial W'_{22}}\cdots\frac{\partial Z_{22}}{\partial W_{21}}\frac{\partial W_{21}}{\partial A_{11}}\frac{\partial A_{11}}{\partial Z_{11}}\frac{\partial Z_{11}}{\partial W_{11}} \tag{6}$$

Because $Z_{11} = W_{11}X$, in which X are input data, as we mentioned before, the gradient for W_{11}: ∂W_{11} will have X as coefficients. **However, distinctively, under the prerequisite training process, those input data X here is the second batch of generated fake data, not real data.**

Therefore, we can derive that: because E (discriminator's loss) in Eq. (7) is differentially private (noised), $\frac{\partial E}{\partial A'_{31}}$ will be differentially private (noised), so as $\frac{\partial A'_{31}}{\partial Z'_{31}}$,..., so as $\frac{\partial Z_{22}}{\partial W_{21}}$,.., so as $\frac{\partial Z_{11}}{\partial W_{11}}$, therefore, the gradient for W_{11}: $\frac{\partial E}{\partial W_{11}}$ is differentially private (noised). What the parameter W_{11} obtains is the differentially private gradient.

Similar to gradient for W_{11}, we can derive that gradients for generator's other parameters are also differentially private.

Until now, we can conclude that under the training process prerequisite, applying differential privacy to discriminator's loss on the second batch of generated fake data, the differential privacy can pass all the way down to generator's gradients, and because the last layer of backpropagation involve only generated fake data, differential privacy can be guaranteed for the whole backpropagation.

In terms of generated fake input data X, the generator might try to leverage well organized X to extract sensitive information from the gradient ∂W_{11}, because the gradient is differentially private, this intent will be handicapped by the noise involved.

As we can see, through propagation effect, because the gradients for the generator are differentially private, the generator and its generated samples will also be differentially pri-

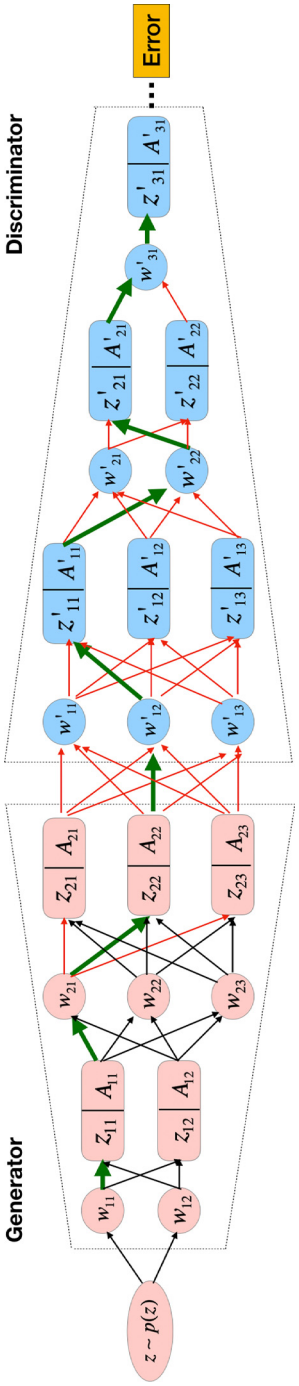

Fig. 2 – An example of detailed GAN structure.

vate. Therefore, applying differential privacy to discriminator's loss can protect privacy of sensitive training datasets.

3.3. *Why applying differential privacy to loss benefits our method?*

In this subsection, we explain why choosing discriminator's loss can benefit our method. As we know, in a GAN, there could be hundreds of thousands of parameters, therefore, the number of gradients for parameters would be very huge. While the dimension of discriminator's loss, which depends on the type of loss function, is usually small. Therefore, the computation workload of applying differential privacy to discriminator's loss will be much less than the computation workload of achieving differentially private gradients. Thus, applying differential privacy to discriminator's loss will be more efficient.

3.4. *How we choose the noise level σ?*

Because the scale of loss g_loss_i will change in every epoch of training, here we utilize an adaptive method to set the bound C.

As we can see, in every training epoch, the generator actually submits two batches of generated fake samples to the discriminator. The first batch is used to train the discriminator, while the second batch is the batch used to compute discriminator's loss *g_loss*.

In our scheme, we use the discriminator's loss on the **first batch** of generated samples $d\_fake \leftarrow D(\text{fake\_data}_t)$ as the scale for the bound C for loss *g_loss*, which is computed on the second batch of generated samples. We set $C = 0.7 * d\_fake$ (here we set a slightly tight bound in our scheme).

As we can see, these two batches of generated fake samples come from the same state of the generator, the loss on the first batch is computed by the discriminator before updating, while the loss on the second batch is computed by the discriminator after updating. These two loss will be close to each other. Therefore, we can use the first batch discriminator's loss as a reference to bound the loss on the second batch. To avoid a loose bond, we add a ratio as 0.7 to the bound. That is because the discriminator's loss on the first batch is, with high probability, a little bit higher than the loss on the second batch. That is because the loss for the second batch is computed by the updated discriminator, which is supposed to be better at classifying samples than the discriminator before updating. So we add a ratio (less than 1) to reduce the bound. Of course, this ratio could be adjusted in different scenarios.

3.5. *Differential privacy proof and privacy loss*

In this subsection, we prove Algorithm 1 in our scheme can be bounded as (ε, δ)-differentially private, then provide the measurement for privacy loss. To compare with other proposed differentially private GANs in privacy loss, we use the same measurement as works DP-GAN (Xie et al., 2018) and dp-GAN (Zhang et al., 2018).

Privacy loss is a random variable dependents on the random noise added to the algorithm. A mechanism $\mathcal{M}$ is (ε, δ)-differentially private is equivalent to a certain tail bound on $\mathcal{M}$'s privacy loss random variable. Similar to works DP-GAN

(Xie et al., 2018) and dp-GAN (Zhang et al., 2018), we also use the moment accountant introduced in Abadi et al. (2016) to keep track of a bound on the moments of the privacy loss random variable (Eq. (7)). The moments accountant can be applied for composing Gaussian mechanisms with random sampling. As shown in Algorithm 1, we update the state by sequentially applying Gaussian differentially private mechanisms during training the discriminator.

3.6. *Differential privacy proof*

Proof. We compute the log moments of the privacy loss random variable, which compose linearly. Then combine the moments bounds with the standard Markov inequality to obtain the tail bound, which is the privacy loss in the sense of differentially privacy.

For neighboring databases $d, d' \in \mathcal{D}^n$, a mechanism $\mathcal{M}$, auxiliary input aux, and an outcome $o \in \mathcal{R}$, define the privacy loss at o as:

$$c(o; \mathcal{M}, \text{aux}, d, d') \triangleq \log \frac{\Pr[\mathcal{M}(\text{aux}, d)=o]}{\Pr[\mathcal{M}(\text{aux}, d')=o]} \tag{7}$$

As shown in Eq. (7), this is an instance of adaptive composition, which we let the auxiliary input aux of the k^{th} mechanism $\mathcal{M}_k$ be the output of all the previous mechanisms.

For a given mechanism denoted as $\mathcal{M}$, we define the λ^{th} moment $\alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d')$ as the log of the moment generating function evaluated at the value λ:

$$\alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d') \triangleq \log \mathbb{E}_{o\sim\mathcal{M}(\text{aux},d)}[\exp(\lambda c(o; \mathcal{M}, \text{aux}, d, d'))]$$

Then we define bound for all possible $\alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d')$ as:

$$\alpha_{\mathcal{M}}(\lambda) \triangleq \max_{\text{aux}, d, d'} \alpha_{\mathcal{M}}(\lambda; \text{aux}, d, d') \tag{8}$$

Which can be used to prove privacy guarantees of a mechanism. The maximum is calculate over all possible aux and all the neighboring databases d, d'.

We recall and use some properties of α introduced and proved in Abadi et al. (2016) to prove the bound.

Theorem 1. *Let $\alpha_{\mathcal{M}}$ defined as above, then*

*1. **[Composability]** Suppose that a mechanism $\mathcal{M}$ consists of a sequence of adaptive mechanisms $\mathcal{M}_1, ..., \mathcal{M}_k$, where $\mathcal{M}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$. Then, for any λ,*

$$\alpha_{\mathcal{M}}(\lambda) \le \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(\lambda) \tag{9}$$

*2. **[Tail bound]** For any $\varepsilon > 0$, the mechanism $\mathcal{M}$ is (ε, δ)-differentially private (we use a common value for δ as 10^{-5}) for*

$$\delta = \min_{\lambda} \exp(\alpha_{\mathcal{M}}(\lambda) - \lambda\varepsilon) \tag{10}$$

Now we need to prove the bound for every step value $\alpha_{\mathcal{M}_t}(\lambda)$. To compute the bound, we need some parameters as follows:

- We use $f(\cdot)$ to denote the function applying Gaussian differentially private mechanism, which is the loss g_loss_i in Algorithm 1, we clip the loss function $f(\cdot)$ to a threshold C, therefore we have $\|f(\cdot)\|_2 \le C$.
- σ is the standard variance of Gaussian distribution, in Algorithm 1, $\sigma \ge 1$
- Sampling probability from the training dataset is denoted as q, in Algorithm 1, the sampling probability is S/N, in which S is the batch size of real samples; N is the total number of samples of training dataset.
- Training epochs (steps) in Algorithm 1 is T.

With these settings, we can obtain a bound for $\alpha_{\mathcal{M}}(\lambda)$, we use a theorem in work (Abadi et al., 2016) to facilitate our proof.

Theorem 2. *Suppose that $f : \mathcal{D} \to \mathbb{R}^p$, with $\|f(\cdot)\|_2 \le C$. Let $\sigma \ge 1$ and let $\mathcal{J}$ be a sample from $[n]$, where each $i \in [n]$ is chosen independently with probability $q < \frac{1}{16\sigma}$. Then for any positive integer $\lambda \le \sigma^2 \ln \frac{1}{q\sigma}$, the mechanism $\mathcal{M}_t(d) = \sum_{i\in J} f(d_i) + \mathcal{N}(0, C^2\sigma^2)$ satisfies*

$$\alpha_{\mathcal{M}_t}(\lambda) \le \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3\lambda^3/\sigma^3) \tag{11}$$

The proof for Theorem 2 is attached in Appendix.

With T steps composed, we can obtain a whole bound for Algorithm 1.

$$\alpha_{\mathcal{M}}(\lambda) \le T\left(\frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3\lambda^3/\sigma^3)\right) \tag{12}$$

□

3.7. *Privacy loss*

With the bound for $\alpha_{\mathcal{M}}(\lambda)$, we can derive a measurement of the differential privacy parameters (ε, δ) for Algorithm 1. With $\lambda \le \sigma^2 \ln(1/q\sigma)$, $q < \frac{1}{16\sigma}$,

$$\alpha_{\mathcal{M}}(\lambda) \le \frac{Tq^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}\left(\frac{Tq^2\lambda^2 \frac{q\lambda}{\sigma}}{\sigma^2}\right) \tag{13}$$

We can see the bound for $\alpha_{\mathcal{M}}(\lambda)$ is dominated by $\frac{Tq^2\lambda(\lambda+1)}{(1-q)\sigma^2}$. We set a slightly tight bound for $\alpha_{\mathcal{M}}(\lambda)$ for easy following reduction as:

$$\alpha_{\mathcal{M}}(\lambda) \le Tq^2\lambda^2/\sigma^2 \tag{14}$$

Then combine with Theorem 1.1 [Tail bound], for any $\varepsilon < 10$ (we set the differential privacy loss ε threshold not exceed 10 for our scheme and set a slightly tight bound for easy following reduction), we can have:

$$Tq^2\lambda^2/\sigma^2 \le \lambda\varepsilon/2 \tag{15}$$

$$\exp(-\lambda\varepsilon/2) \le \delta \tag{16}$$

According to these two inequality, and $\lambda > 1, \sigma > 1$, we can have:

$$Tq^2\lambda\varepsilon/\sigma^2 \le \varepsilon^2/2 \tag{17}$$

$$\lambda\varepsilon \geq 2\ln\frac{1}{\delta} \tag{18}$$

Then we set the bound for ε, which is the relation between σ and ε.

$$\varepsilon = 2q\sqrt{T\ln(\frac{1}{\delta})}/\sigma \tag{19}$$

With this evaluation method for ε, we can evaluate the bound (ε, δ) and claim our Algorithm 1 is (ε, δ)-differentially private.

4. Evaluation

4.1. Datasets

We evaluate the performance of our scheme on MNIST, Fashion MNIST, and SVHN datasets. MNIST is a 10-class handwritten digit recognition dataset consisting of 60,000 training examples and 10,000 test examples (LeCun et al., 1998), each example is a 28×28 size greyscale image. Similarly, Fashion MNIST is a 10-class dataset of fashion images, also consisting of 60,000 training examples and 10,000 testing examples (Xiao et al., 2017), each example is a 28×28 size gray-level image. MNIST (produced in 1998) has been as a benchmark for machine learning and data science algorithms for years, and now Fashion MNIST (produced in 2017) serves as a replacement for the MNIST dataset for benchmarking machine learning algorithms. SVHN (produced in 2011) is obtained from house numbers in Google Street View images. It is a 10 class 32×32 RGB digit image datasets (Netzer et al., 2011), that can be seen as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images, 73,257 digits for training, 26,032 digits for testing, and 531,131 additional, somewhat less difficult samples, to use as extra training data) and comes from a significantly harder, real-world problem (recognizing digits and numbers in natural scene images).

4.2. Implementation

In our experiment, we use deep convolutional generative adversarial network (DCGAN) as GAN structure, which is a good fit for image-based test datasets. Notice that, our method is not tailored to DCGANs, other types of GANs using the prerequisite training process can use our method to achieve differential privacy.

We implement our scheme as described in Algorithm 1, utilize DCGAN as GAN structure. In Fig. 3 and Fig. 4, we show an example of structures of a discriminator and a generator used in our experiment for MNIST dataset, they use deep convolutional network architectures.

We program our codes in Python, execute them on Google Colab[2], which provides free access to online GPU and TPU. We use Tensorflow[3] and Keras[4] as backend. We will open source our codes along with this paper.

We train the DCGAN for 10,000 epochs, batch size S in Algorithm 1 for MNIST and Fashion MNIST is 600, latent dimension for generator is 100. For SVHN, the batch size is 700, and the epoch is 10,000.

As we expect, choosing loss function as the target of applying differential privacy benefits our experiments. It is easy to program and suitable for different GAN structures. Also, there is no need to modify core functions of Tensorflow or Keras to implement our scheme. Because most of GANs use optimizer functions provided by Tensorflow to compute gradients and update parameters, applying differential privacy needs to modify the Tensorflow Library.

In Fig. 5, we show some generated fake samples for MNIST, Fashion MNIST, and SVHN from our differentially private generators, in which noise level $\sigma = 1.0$. As shown in Fig. 5, generated fake data looks real and like data comes from the real world rather than some synthetic data generated from noise.

As shown in Fig. 5, generated samples look like real samples from MNIST, Fashion MNIST, and SVHN. For example, generated synthetic fake Fashion MNIST samples look like real fashion items come from the real world, so as generated synthetic fake MNIST and SVHN samples.

Notice that, generated fake samples looks a little bit blurry compared to real samples. The main reasons come from two aspects: Firstly, GANs can not perfectly simulate real data's distributions, the quality of generated samples depend on the GAN structure, the number of training epochs, and the training optimizer, etc. Secondly, the noise added to discriminator's loss will to some extent affect the ability of the generator capturing features of real training data.

We also evaluate our scheme under different Gaussian noise levels σ, then compute privacy losses accordingly. With noise level σ and differential privacy parameter δ, we can calculate the privacy loss ε according to Eq. (19). Some other parameters are as follows: batch size S for MNIST, Fashion MNIST is 600, for SVHN is 700. Training epoch T for MNIST, Fashion MNIST, and SVHN is 10,000. Total numbers of training samples N in MNIST and Fashion MNIST is 60,000; for SVHN, there are 73,257 digits for training and 26,032 digits for testing. We list the related experimental results in Table 1. **As seen from** Table 1, our experiments achieve low privacy losses; with such low privacy losses, we still get satisfying accuracy for MNIST and good accuracy for Fashion MNIST and SVHN, which are very surprising results.

As seen from Table 1, with more noise added, we can achieve stronger privacy protection for training datasets, which means lower privacy loss. For example, with noise level $\sigma = 1.0$, we achieve privacy loss as $(\varepsilon, \delta) = (6.786, 10^{-5})$ for MNIST and Fashion MNIST, $(\varepsilon, \delta) = (6.484, 10^{-5})$ for SVHN. With noise level $\sigma = 3.0$, we obtain $(\varepsilon, \delta) = (2.262, 10^{-5})$ privacy loss for MNIST and Fashion MNIST and $(\varepsilon, \delta) = (2.161, 10^{-5})$ for SVHN.

To test the quality of generated data from our differentially private generators, we use generated data as training datasets

[2] https://colab.research.google.com.

[3] https://www.tensorflow.org.

[4] https://keras.io.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 14, 14, 32)	320
leaky_re_lu_1 (LeakyReLU)	(None, 14, 14, 32)	0
dropout_1 (Dropout)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 7, 7, 64)	18496
zero_padding2d_1 (ZeroPaddin	(None, 8, 8, 64)	0
batch_normalization_1 (Batch	(None, 8, 8, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
conv2d_3 (Conv2D)	(None, 4, 4, 128)	73856
batch_normalization_2 (Batch	(None, 4, 4, 128)	512
leaky_re_lu_3 (LeakyReLU)	(None, 4, 4, 128)	0
dropout_3 (Dropout)	(None, 4, 4, 128)	0
conv2d_4 (Conv2D)	(None, 4, 4, 256)	295168
batch_normalization_3 (Batch	(None, 4, 4, 256)	1024
leaky_re_lu_4 (LeakyReLU)	(None, 4, 4, 256)	0
dropout_4 (Dropout)	(None, 4, 4, 256)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_1 (Dense)	(None, 1)	4097

Fig. 3 – Structure of a discriminator.

to train deep learning models for MNIST, Fashion MNIST, and SVHN, then test the deep learning models on real test samples. We list the test accuracies in Table 1. With more noise added to differentially private GANs, generated samples will result in lower accuracies. With Gaussian noise $\sigma = 1.0$, privacy loss $\varepsilon = 6.786$, generated MNIST samples and Fashion MNIST samples achieve 98.03% and 85.24% accuracies respectively; with Gaussian noise $\sigma = 1.0$, privacy loss $\varepsilon = 6.484$, generated SVHN samples achieve 88.65% accuracy. When the noise level is set as 5.0, with privacy loss $\varepsilon = 1.357$, generated MNIST and Fashion MNIST samples achieve 97.20% and 83.49% accuracies respectively, with privacy loss $(\varepsilon, \delta) = (1.297, 10^{-5})$, generated SVHN samples achieve 87.22% accuracy.

As we can expect, there is always a tradeoff between privacy and performance. Because we use differential privacy to provide protection for sensitive training datasets, generated samples achieve slightly lower accuracies than baselines. As shown in Table 1, with privacy loss $\varepsilon = 6.786$, generated MNIST samples achieve 98.03% accuracy, 1.17% lower than the baseline 99.20%; with baseline for Fashion MNIST 92.40%, generated Fashion MNIST samples achieve 85.24% accuracy, 7.16% lower than the baseline; with privacy loss $\varepsilon = 6.484$, generated SVHN samples achieve 88.65% accuracy, 8.09% less than the baseline 96.74%. These inferior performances mainly due to two reasons: firstly, we add noise to GANs to achieve differential privacy, the added noise will to some extent affect the quality of generated data, thus affecting the accuracies of deep learning models it achieved; Secondly, we use generated fake samples to train deep learning models, but test them on real samples, this gap causes some bias of accuracies as well.

Because dataset Fashion MNIST and SVHN datasets are considered as more complex than MNIST, our experiments ob-

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 6272)	633472
reshape_1 (Reshape)	(None, 7, 7, 128)	0
up_sampling2d_1 (UpSampling2	(None, 14, 14, 128)	0
conv2d_5 (Conv2D)	(None, 14, 14, 128)	147584
batch_normalization_4 (Batch	(None, 14, 14, 128)	512
activation_1 (Activation)	(None, 14, 14, 128)	0
up_sampling2d_2 (UpSampling2	(None, 28, 28, 128)	0
conv2d_6 (Conv2D)	(None, 28, 28, 64)	73792
batch_normalization_5 (Batch	(None, 28, 28, 64)	256
activation_2 (Activation)	(None, 28, 28, 64)	0
conv2d_7 (Conv2D)	(None, 28, 28, 1)	577
activation_3 (Activation)	(None, 28, 28, 1)	0

Fig. 4 – Structure of a generator.

Table 1 – Accuracies achieved by generated fake data and privacy loss under different noise levels.

Dataset	Noise level	Privacy loss	Accuracy	Baseline
MNIST	$\sigma = 1.0$	$(\varepsilon, \delta) = (6.786, 10^{-5})$	98.03%	99.20%
	$\sigma = 1.5$	$(\varepsilon, \delta) = (4.524, 10^{-5})$	97.86%	
	$\sigma = 3.0$	$(\varepsilon, \delta) = (2.262, 10^{-5})$	97.57%	
	$\sigma = 5.0$	$(\varepsilon, \delta) = (1.357, 10^{-5})$	97.20%	
Fashion MNIST	$\sigma = 1.0$	$(\varepsilon, \delta) = (6.786, 10^{-5})$	85.24%	92.40%
	$\sigma = 1.5$	$(\varepsilon, \delta) = (4.524, 10^{-5})$	85.05%	
	$\sigma = 3.0$	$(\varepsilon, \delta) = (2.262, 10^{-5})$	84.64%	
	$\sigma = 5.0$	$(\varepsilon, \delta) = (1.357, 10^{-5})$	83.49%	
SVHN	$\sigma = 1.0$	$(\varepsilon, \delta) = (6.484, 10^{-5})$	88.65%	96.74%
	$\sigma = 1.5$	$(\varepsilon, \delta) = (4.323, 10^{-5})$	88.43%	
	$\sigma = 3.0$	$(\varepsilon, \delta) = (2.161, 10^{-5})$	87.88%	
	$\sigma = 5.0$	$(\varepsilon, \delta) = (1.297, 10^{-5})$	87.22%	

tain inferior performances on Fashion MNIST and SVHN than MNIST.

In conclusion, according to these experiments, our method can achieve high-quality differentially private generators, providing strong privacy protection for training datasets with low privacy loss.

5. Comparison with related works

In this section, we illustrate some state-of-the-art works related to our study and compare our method with some of these related works.

DP-GAN (Xie et al., 2018), dp-GAN (Zhang et al., 2018), and DP-CGAN (Torkzadehmahani et al., 2019) use differentially private gradient descent (DP-SGD) or modified DP-SGD to achieve differentially private GANs. PATE-GAN (Jordon et al., 2018) is designed under PATE architecture (Papernot et al., 2016), uses multiple teacher models and applies differential privacy on majority voting labelling, then combines with a simple classifier to achieve a differentially private GAN. G-PATE (Long et al., 2020) trains discriminators under differential privacy using PATE architecture (Papernot et al., 2016), then average the gradients for a global generator to update its parameters.

DP-GAN (Xie et al., 2018), dp-GAN (Zhang et al., 2018), DP-CGAN (Torkzadehmahani et al., 2019) directly or indirectly use differential privacy on gradients, which need to use a modified Tensorflow library (Tensorflow Privacy). Using this modified Tensorflow library can be extremely inefficient. According to DP-CGAN (Torkzadehmahani et al., 2019)'s open codes[5], training a differentially private conditional GAN on MNIST can take over two hours on Colab's TPU. While our scheme needs no

[5] https://colab.research.google.com/github/ricardocarvalhods/dpcgan/blob/master/DP_CGAN_MNIST.ipynb.

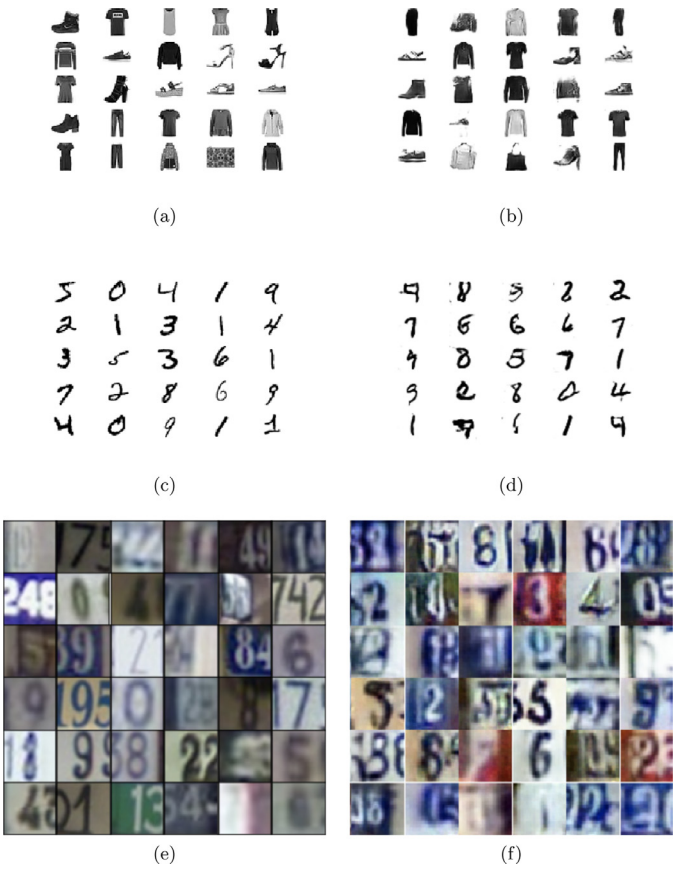

Fig. 5 – Real (a) and fake (b) samples of Fashion MNIST; Real (c) and fake (d) samples of MNIST; Real (e) and fake (f) samples of SVHN; Fake samples are generated from differentially private DCGANs.

modifications on Tensorflow's core functions, and will bring high efficiency.

Due to lack of resources of other works' open codes, we only compare our method with DP-CGAN (Torkzadehmahani et al., 2019). We modify their open codes to apply our scheme, replace the Tensorflow Privacy Library by Tensorflow, then apply differential privacy to discriminator's loss. Experimental results show that based on the same GAN structure, our scheme uses only 252 s (averagely), compared with DP-CGAN (Torkzadehmahani et al., 2019), which takes **2 hours 6 min 43 s (7603 s)**, the running time of our scheme is only 3.3% of DP-CGAN (Torkzadehmahani et al., 2019)'s running time. Notice that, our scheme also achieves slightly higher accuracy (89.07%) than DP-CGAN (Torkzadehmahani et al., 2019) (88.16%). We list the detail numbers in Table 2.

Because our scheme has an advantage in execution efficiency, we can expect that the training time for a deeper differentially private GAN achieving higher accuracy will also be acceptable. As we can imagine, the running time of training a deeper differentially private GAN using DP-CGAN (Torkzadehmahani et al., 2019) will be much longer than it takes on a simple CGAN.

As we demonstrated in Section Evaluation, we achieved a differentially private DCGAN, compared with DP-CGAN (Torkzadehmahani et al., 2019), our differentially private DCGAN has a deeper structure and achieves higher accuracy. We list the running time, privacy loss and accuracy comparison in Table 3. Due to TPU memory size limit on Google Colab, we failed to train a differentially private DCGAN with the same structure as our scheme using DP-CGAN (Torkzadehmahani et al., 2019)'s method. Therefore, we only compare our differentially private DCGAN with DP-CGAN (Torkzadehmahani et al., 2019).

With lower privacy loss, our scheme achieves higher accuracy using much less time than DP-CGAN (Torkzadehmahani et al., 2019). With privacy loss 6.786, achieving 98.03% accuracy, our method needs **4 min 36 s (276 s)** during training the differentially private DCGAN, while

Table 2 – Comparison of running time while training a differentially private CGAN using our method and DP-CGAN (Torkzadehmahani et al., 2019).

Dataset	Scheme	Accuracy	Time (s)
MNIST	DP-CGAN (Torkzadehmahani et al., 2019)	88.16%	7603
	Our scheme	**89.07%**	**252**

Table 3 – Comparison of running time during training differentially private GANs using our method and DP-CGAN (Torkzadehmahani et al., 2019).

Dataset	Scheme	Privacy loss ε	Accuracy	Non-DP Time (s)	Time (s)	Increase
MNIST	DP-CGAN (Torkzadehmahani et al., 2019)	9.6	88.16%	165	7603	4507.9% ↑
	Our scheme	**6.786**	**98.03%**	202	**276**	**36.6%** ↑

Table 4 – Comparison of accuracies among three related works and ours.

Dataset	Scheme	Privacy loss ε	Accuracy
MNIST	DP-GAN (Xie et al., 2018)	11.5	99.00%
	dp-GAN (Zhang et al., 2018)	4.0	86.40%
	DP-CGAN (Torkzadehmahani et al., 2019)	9.6	88.16%
	G-PATE (Long et al., 2020)	10	80.92%
	Our scheme	**6.786**	**98.03%**

work (Torkzadehmahani et al., 2019) (DP-CGAN) takes **2 hours 6 min 43 s (7603 s)** to train a differentially private CGAN, achieving 88.16% accuracy with privacy loss 9.6. These experimental results show that training a deeper differentially private GAN, achieving higher accuracies than DP-CGAN (Torkzadehmahani et al., 2019), our scheme needs much less time than DP-CGAN (Torkzadehmahani et al., 2019), indicates our scheme can bring high efficiency.

We also measured the running time of GANs in DP-CGAN (Torkzadehmahani et al., 2019) and ours when not using differential privacy, these numbers can help us to measure how much time has been increased due to applying differential privacy. As we can see from Table 3, DP-SGD method used in DP-CGAN (Torkzadehmahani et al., 2019) increases 4507.9% time compared with training the GAN without using differential privacy, while our method only brings 36.6% increase to the training time.

We compared the accuracies achieved by generated samples from differentially private GANs of our work and other four works: DP-GAN (Xie et al., 2018), dp-GAN (Zhang et al., 2018), DP-CGAN (Torkzadehmahani et al., 2019) and G-PATE (Long et al., 2020) mentioned above. The comparison is list in Table 4. Due to lack of experimental results on Fashion MNIST from those three related works, we only list accuracies tested on MNIST dataset.

As shown from Table 3, with less privacy loss, generated data from our differentially private GAN achieves higher accuracy than works (Long et al., 2020; Torkzadehmahani et al., 2019; Zhang et al., 2018). With higher privacy loss, work (Xie et al., 2018) can achieve slightly higher accuracy than our scheme. However, since work (Xie et al., 2018) is designed based on DP-SGD algorithm, our method will win in efficiency.

In conclusion, compared with state-of-the-art related works, our method has advantages in efficiency, privacy loss, and accuracy.

6. Conclusion

Motivated by developing an efficient method for differentially private GAN, we start from a new perspective, use discriminator's loss as vehicle to apply differential privacy. We implement our method and test its performance on three real-world datasets, experimental results show that our method can achieve high-quality differentially private GANs with low privacy loss. Generated data from our differentially private GANs can achieve close accuracies as real training datasets. Additionally, because there is no need to modify Tensorflow library, our method is easy to implement and highly efficient, making it outstanding among existing methods. With these advantages, we believe our method can be widely applied in the very near future.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Chunling Han: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing. **Rui Xue:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing.

Acknowledgements

The authors are supported by National Natural Science Foundation of China (No. 61772514), National Key R&D Program of China (No. 2017YFB1400700). We would like to thank Professor Yan Huang and Dr. Haoai Zhao from Indiana University Bloomington for their substantial support, insightful comments and suggestions; Dr. Rui Zhang from Chinese Academy of Sciences for her discussion and guidance. Special thanks goes to Han's family and Xue's family for their understanding and support.

Appendix A

Theorem 3. *Suppose that $f : \mathcal{D} \to \mathbb{R}^p$, with $\|f(\cdot)\|_2 \le C$. Let $\sigma \ge 1$ and let $\mathcal{J}$ be a sample from $[n]$, where each $i \in [n]$ is chosen independently with probability $q < \frac{1}{16\sigma}$. Then for any positive integer $\lambda \le \sigma^2 \ln\frac{1}{q\sigma}$, the mechanism $\mathcal{M}_t(d) = \sum_{i\in J} f(d_i) + \mathcal{N}(0, C^2\sigma^2)$ satisfies*

$$\alpha_{\mathcal{M}_t}(\lambda) \le \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + \mathcal{O}(q^3\lambda^3/\sigma^3) \tag{20}$$

Proof. Fix d' and let $d = d' \cup \{d_n\}$. Without loss of generality, $f(d_n) = \vec{e_1}$ and $\sum_{i\in J\setminus[n]} f(d_i) = \vec{0}$. Thus $\mathcal{M}_t(d)$ and $\mathcal{M}_t(d')$ are distributed identically except for the first coordinate. Therefore, we have a one-dimensional problem. Let μ_0 denote the probability density function of $\mathcal{N}(0, \sigma^2)$ and let μ_1 denote the probability density function of $\mathcal{N}(1, \sigma^2)$. Thus:

$$\begin{aligned}&\mathcal{M}_t(d') \sim \mu_0\\ &\mathcal{M}_t(d) \sim \mu \triangleq (1-q)\mu_0 + q\mu_1\end{aligned} \tag{21}$$

We need these two inequality

$$\mathbb{E}_{z\sim\mu}[(\mu(z)/\mu_0(z))^\lambda] \le \alpha \tag{22}$$

$$\mathbb{E}_{z\sim\mu_0}[(\mu_0(z)/\mu(z))^\lambda] \le \alpha \tag{23}$$

the parameter α will be determined later.

We will use the same method to prove those two inequality above. Assume we have two distributions ν_0 and ν_1, and we need to bound

$$\mathbb{E}_{z\sim\nu_0}[(\nu_0(z)/\nu_1(z))^\lambda] = \mathbb{E}_{z\sim\nu_1}[(\nu_0(z)/\nu_1(z))^{\lambda+1}] \tag{24}$$

By using binomial expansion, we have

$$\begin{aligned}\mathbb{E}_{z\sim\nu_1}[(\nu_0(z)/\nu_1(z))^{\lambda+1}] &= \mathbb{E}_{z\sim\nu_1}[(1+(\nu_0(z)-\nu_1(z))/\nu_1(z))^{\lambda+1}]\\ &= \sum_{t=0}^{\lambda+1}\binom{\lambda+1}{t}\mathbb{E}_{z\sim\nu_1}\\ &\quad\times[((\nu_0(z)-\nu_1(z))/\nu_1(z))^{\lambda+1}]\end{aligned} \tag{25}$$

The first term in Eq. (25) is 1, and the second term is

$$\begin{aligned}\mathbb{E}_{z\sim\nu_1}\left[\frac{\nu_0(z)-\nu_1(z)}{\nu_1(z)}\right] &= \int_{-\infty}^{\infty}\nu_1(z)\frac{\nu_0(z)-\nu_1(z)}{\nu_1(z)}dz\\ &= \int_{-\infty}^{\infty}\nu_0(z)dz - \int_{-\infty}^{\infty}\nu_1(z)dz\\ &= 1-1=0\end{aligned} \tag{26}$$

To prove Theorem 3, it can be seen that for both $\nu_0 = \mu$, $\nu_1 = \mu_0$ and $\nu_0 = \mu_0$, $\nu_1 = \mu$, the third term is bounded by $q^2\lambda(\lambda+1)/(1-q)\sigma^2$ and dominating the sum of the remaining terms. We take ($\nu_0 = \mu_0$, $\nu_1 = \mu$) as an example to prove, similarly, the other case can be proved in the same way.

To compute the upper bound for the third term in Eq. (25), we note that $\mu(z) \ge (1-q)\mu_0(z)$, and we have

$$\begin{aligned}\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^2\right] &= q^2\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu(z)}\right)^2\right]\\ &= q^2\int_{-\infty}^{\infty}\frac{(\mu_0(z)-\mu_1(z))^2}{\mu(z)}dz\\ &\le \frac{q^2}{1-q}\int_{-\infty}^{\infty}\frac{(\mu_0(z)-\mu_1(z))^2}{\mu_0(z)}dz\\ &= \frac{q^2}{1-q}\mathbb{E}_{z\sim\mu_0}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^2\right]\end{aligned} \tag{27}$$

As the fact that for any $a \in \mathbb{R}$, $\mathbb{E}_{z\sim\mu_0}\exp(2az/2\sigma^2) = \exp(a^2/2\sigma^2)$. We have

$$\begin{aligned}\mathbb{E}_{z\sim\mu_0}\left[\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^2\right] &= \mathbb{E}_{z\sim\mu_0}\left[\left(1-\exp(\frac{2z-1}{2\sigma^2})\right)^2\right]\\ &= 1-2\mathbb{E}_{z\sim\mu_0}\left[\exp(\frac{2z-1}{2\sigma^2})\right] + \mathbb{E}_{z\sim\mu_0}\left[\exp(\frac{4z-2}{2\sigma^2})\right]\\ &= 1-2\exp(\frac{1}{2\sigma^2})\cdot\exp(\frac{-1}{2\sigma^2}) + \exp(\frac{4}{2\sigma^2})\cdot\exp(\frac{-2}{2\sigma^2})\\ &= \exp(\frac{1}{\sigma^2})-1\end{aligned} \tag{28}$$

Therefore, the third term in the binomial expansion (Eq. (25))

$$\binom{1+\lambda}{2}\mathbb{E}_{z\in\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^2\right] \le \frac{\lambda(\lambda+1)q^2}{(1-q)\sigma^2} \tag{29}$$

We use standard calculus to bound the remaining terms:

$$\forall z \le 0 : |\mu_0(z)-\mu_1(z)| \le -(z-1)\mu_0(z)/\sigma^2 \tag{30}$$

$$\forall z \ge 1 : |\mu_0(z)-\mu_1(z)| \le z\mu_1(z)/\sigma^2 \tag{31}$$

$$\forall 0 \le z \le 1 : |\mu_0(z)-\mu_1(z)| \le \mu_0(z)(\exp(1/2\sigma^2)-1) \le \mu_0(z)/\sigma^2 \tag{32}$$

Then we can get

$$\begin{aligned}\mathbb{E}_{z\sim\mu}\left[\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right] &\le \int_{-\infty}^{0}\mu(z)\left|\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right|dz\\ &\quad+\int_{0}^{1}\mu(z)\left|\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right|dz\\ &\quad+\int_{1}^{\infty}\mu(z)\left|\left(\frac{\mu_0(z)-\mu(z)}{\mu(z)}\right)^t\right|dz\end{aligned} \tag{33}$$

We consider these terms individually. By using these three observations: (1) $\mu_0 - \mu = q(\mu_0 - \mu_1)$, (2)$\mu \ge (1-q)\mu_0$ and (3) $\mathbb{E}_{\mu_0}[|z|^t] \le \sigma^t(t-1)!!$. As we can see, the first term can be bounded by

$$\frac{q^t}{(1-q)^{t-1}\sigma^{2t}}\int_{-\infty}^{0}\mu_0(z)|z-1|^t dz \le \frac{(2q)^t(t-1)!!}{2(1-q)^{t-1}\sigma^t} \tag{34}$$

The upper bound for the second term is

$$\begin{aligned}\frac{q^t}{(1-q)^t}\int_0^1\mu(z)\left|\left(\frac{\mu_0(z)-\mu_1(z)}{\mu_0(z)}\right)^t\right|dz &\le \frac{q^t}{(1-q)^t}\int_0^1\mu(z)\frac{1}{\sigma^{2t}}dz\\ &\le \frac{q^t}{(1-q)^t\sigma^{2t}}\end{aligned} \tag{35}$$

The upper bound for the third term is

$$\begin{aligned}
&\frac{q^t}{(1-q)^{t-1}\sigma^{2t}} \int_1^{\infty} \mu_0(z)\left(\frac{z\mu_1(z)}{\mu_0(z)}\right)^t dz \\
&\leq \frac{q^t}{(1-q)^{t-1}\sigma^{2t}} \int_1^{\infty} \mu_0(z)\exp((2tz-t)/2\sigma^2)z^t dz \\
&\leq \frac{q^t\exp((t^2-t)/2\sigma^2)}{(1-q)^{t-1}\sigma^{2t}} \int_0^{\infty} \mu_0(z-t)z^t dz \\
&\leq \frac{(2q)^t\exp((t^2-t)/2\sigma^2)(\sigma^t(t-1)!! + t^t)}{2(1-q)^{t-1}\sigma^{2t}} \qquad (36)
\end{aligned}$$

According to the definition and range of q, σ and λ, we can see that the three terms, and their sum, drop off geometrically fast in t for $t > 3$. Therefore, the binomial expansion (Eq. (25)) is dominated by the $t = 3$ term, which is $\mathcal{O}(q^3\lambda^3/\sigma^3)$. Therefore, we can claim Theorem 3 is sound. □

REFERENCES

Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM; 2016. p. 308–18.

Augenstein S, McMahan HB, Ramage D, Ramaswamy S, Kairouz P, Chen M, Mathews R, et al. Generative models for effective ML on private, decentralized datasets 2019 arXiv:191106679.

Beaulieu-Jones BK, Wu ZS, Williams C, Lee R, Bhavnani SP, Byrd JB, Greene CS. Privacy-preserving generative deep neural networks support clinical data sharing. Circulation: Cardiovascul. Q. Outcome. 2019;12(7):e005122.

Dwork C, Lei J. Differential privacy and robust statistics, volume 9; 2009. p. 371–80.

Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. Springer; 2006. p. 265–84.

Dwork C, Rothblum GN. Concentrated differential privacy 2016 arXiv:160301887.

Dwork C, Rothblum GN, Vadhan S. Boosting and differential privacy. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. IEEE; 2010. p. 51–60.

Frigerio L, de Oliveira AS, Gomez L, Duverger P. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In: IFIP International Conference on ICT Systems Security and Privacy Protection. Springer; 2019. p. 151–64.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Advances in neural information processing systems; 2014. p. 2672–80.

Jordon J, Yoon J, van der Schaar M. Pate-gan: Generating synthetic data with differential privacy guarantees 2018.

LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc. IEEE 1998;86(11):2278–324.

Long Y, Lin S, Yang Z, Gunter CA, Liu H, Li B. Scalable differentially private data generation via private aggregation of teacher ensembles. 2020. https://openreview.net/forum?id=Hkl6i0EFPH.

Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY. Reading digits in natural images with unsupervised feature learning 2011.

Papernot N, Abadi M, Erlingsson U, Goodfellow I, Talwar K. Semi-supervised knowledge transfer for deep learning from private training data 2016 arXiv:161005755.

Papernot N, Song S, Mironov I, Raghunathan A, Talwar K, Erlingsson U. Scalable private learning with pate 2018 arXiv:180208908.

Torkzadehmahani R, Kairouz P, Paten B. Dp-cgan: Differentially private synthetic data and label generation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019. 0–0

Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms 2017 arXiv:170807747.

Xie L, Lin K, Wang S, Wang F, Zhou J. Differentially private generative adversarial network 2018 arXiv:180206739.

Xu C, Ren J, Zhang D, Zhang Y, Qin Z, Ren K. Ganobfuscator: Mitigating information leakage under gan via differential privacy. IEEE Trans. Inf. Forensic. Secur. 2019;14(9):2358–71.

Zhang X, Ji S, Wang T. Differentially private releasing via deep generative model (technical report) 2018 arXiv:180101594.

Chunling Han currently is a joint Ph.D student at Luddy School of Informatics, Computing and Engineering, in Indiana University Bloomington, IN, USA. She studied in Indiana University Bloomington since September 2018. Before that, she was a Ph.D candidate majored in Cyber Security in State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, CAS, School of Cyber Security, University of Chinese Academy of Sciences, Beijing China, since 2015. She received a bachelor's degree in 2015, from China University of Geosciences in Wuhan China, majored in Information Security. She has six years of research experience in information security, published several academic papers in area of data privacy, privacy-preserving machine learning, Android malware detection and Steganalysis. Her interests broadly lie in data privacy, privacy-preserving machine learning, cryptography, etc.

Rui Xue is currently a researcher and doctoral supervisor in State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, School of Cyber Security in University of Chinese Academy of Sciences, Beijing, China. He received his M.S and Ph.D degrees from Department of Mathematics at Beijing Normal University, in 1988 and 1999. He has decades of research experience in cryptography, cloud data security, and security protocols, has published many academic papers in the area of computer and information security. His research interests include cryptography, security protocols, cloud data security and multimedia security.