Sumit Mukherjee, Yixi Xu, Anusua Trivedi, Nabajyoti Patowary, and Juan L. Ferres

privGAN: Protecting GANs from membership inference attacks at low cost to utility

Abstract: Generative Adversarial Networks (GANs) have made releasing of synthetic images a viable approach to share data without releasing the original dataset. It has been shown that such synthetic data can be used for a variety of downstream tasks such as training classifiers that would otherwise require the original dataset to be shared. However, recent work has shown that the GAN models and their synthetically generated data can be used to infer the training set membership by an adversary who has access to the entire dataset and some auxiliary information. Current approaches to mitigate this problem (such as DPGAN [1]) lead to dramatically poorer generated sample quality than the original non–private GANs. Here we develop a new GAN architecture (privGAN), where the generator is trained not only to cheat the discriminator but also to defend membership inference attacks. The new mechanism is shown to empirically provide protection against this mode of attack while leading to negligible loss in downstream performances. In addition, our algorithm has been shown to explicitly prevent memorization of the training set, which explains why our protection is so effective. The main contributions of this paper are: i) we propose a novel GAN architecture that can generate synthetic data in a privacy preserving manner with minimal hyperparameter tuning and architecture selection, ii) we provide a theoretical understanding of the optimal solution of the privGAN loss function, iii) we empirically demonstrate the effectiveness of our model against several white and black–box attacks on several benchmark datasets, iv) we empirically demonstrate on three common benchmark datasets that synthetic images generated by privGAN lead to negligible loss in downstream performance when compared against non–private GANs. While we have focused on benchmarking privGAN exclusively on image datasets, the architecture of privGAN is not exclusive to image datasets and can be easily extended to other types of datasets. Repository link: https://github.com/microsoft/privGAN.

Keywords: Membership inference, GANs

DOI 10.2478/popets-2021-0041

Received 2020-11-30; revised 2021-03-15; accepted 2021-03-16.

1 Introduction

Much of the recent progress in machine learning and related areas has been strongly dependent on the open sharing of datasets. A recent study shows that the increase in the number of open datasets in biology has led to a strongly correlated increase in the number of data analysis papers [2]. Moreover, in many specialized application areas, the development of novel algorithms is contingent upon the public availability of relevant data. While the public availability of data is essential for reproducible science, in the case of sensitive data, this poses a possible threat to the privacy of the individuals in the dataset.

One way in which privacy of samples in a dataset can be compromised is through membership inference attacks. Membership inference attacks are adversarial attacks where the goal of the adversary is to infer whether one or more samples are a part of a dataset without having explicit access to the dataset. There has been a lot of work on developing membership inference attacks against predictive machine learning models using their outputs [3–6]. Much of this work has focused on exploiting information leakage due to overfitting in many machine learning models [7]. These approaches have been shown to be extremely effective against common machine learning models and have given rise to machine learning methods that are specifically designed to be resistant to such attacks [8–11].

There has recently been a surge of interest in using synthetic data generated from generative models as a privacy preserving way to share datasets [12–14]. While this is an appealing approach, it has been shown that generative models such as GANs are also prone to mem-

Sumit Mukherjee, Yixi Xu, Anusua Trivedi, Juan L. Ferres: AI for Good Research Lab, Microsoft, USA. E-mail: {summukhe,yixi.xu,antriv,jlavista}@microsoft.com
Nabajyoti Patowary: Microsoft, USA. E-mail: napatowa@microsoft.com

orizing their training set [15]. This has been exploited in several recent papers to explore the vulnerability of generative models to membership inference attacks [16–18]. [16] designed a white–box attack on the released discriminator of a GAN and showed that it can be almost 100% accurate in some cases. They also designed a black–box attack, which is comparatively a lot less accurate. [18] designed Monte–Carlo attacks on the generators which are shown to have high accuracy for set membership inference (defined later) and slightly lower accuracy for instance membership inference.

To address this vulnerability to membership inference attacks, we developed a novel GAN architecture namely priv(ate)GAN to enhance membership privacy: the synthetic data generated by the GAN trained on the training samples should be indistinguishable from those generated by the GAN trained on the other data points from the same distribution. To achieve this, an adversary is trained to attack the model, while the generator is trained to fool both the discriminator and the adversary. We empirically demonstrate the effectiveness of this architecture against attacks described in [16, 18] as well as an oracle attack on the discrminator that we designed. We empirically show that privGAN achieves high membership privacy while not sacrificing the sample quality, compared to the original GANs with identical architecture and hyperparameter settings. Our primary contributions in this paper are: i) proposing a novel privacy preserving GAN architecture which requires **minimal additional hyperparameter selection and no additional architecture choices**, ii) providing a theoretical analysis of the optimal solution to the GAN minimax problem and demonstrating that with large enough sample size the generative model learned with privGAN is identical to a non–private GAN, iii) empirically comparing the performance of our architecture against baselines on different membership inference attacks (both on generators and discriminators), and iv) empirically comparing the sample quality of our generated samples with different baselines.

2 Related works

2.1 Membership inference attacks against machine learning models

A membership attacker aims to infer the membership of samples to a training set. To formally understand the adversary, let us first assume there exists a training set X_{train} drawn from some data distribution D, a machine learning model M trained on X_{train} and an adversary A that has access to samples X_{adv} (also drawn from D). The adversary is assumed to have some query access to M such that, given a sample $x \in X_{adv}$, it can compute some function $Q(M, x)$. The goal of the adversary is to then estimate $Pr(x \in X_{train})$ as a function of $Q(M, x)$. [3] demonstrated one of the first membership attacks against discriminative models with only black-box access to the confidence scores for each input data sample. Their approach involved first training many 'shadow' machine learning models. Then an adversary model was trained to identify whether a sample was in the training set of any of the shadow models. This MIA approach was shown to be surprisingly effective against a variety of CNNs. Since then, there have been many such empirical membership attacks designed against machine learning models. Recently, [7, 19] quantified membership privacy leakage as an adversarial game and showed a connection to the generalization error of machine learning algorithms and differential privacy.

There have been several recent papers proposing successful membership inference adversaries against generative models [16, 18]. Both of these works were motivated by the close connection of information leakage to overfitting. More specifically, the generative models tend to memorize the training samples. The success of such adversaries greatly increases the risk to publish even synthetic datasets. To solve this concern, We propose privGAN, and will show the effectiveness of our method against all of these attacks in Section 6.

2.2 Private GANs

A differentially private algorithm guarantees that the algorithm will yield similar outputs with high probability, no matter whether a sample is included in the training dataset or not [20]. Existing works [1, 21] propose new algorithms guaranteeing differential privacy of the discriminator (here the output is the model parameters) by the introduction of systematic noise during the model optimization steps [22]. This also leads to a differentially private generator as differential privacy is immune to post-processing [20]. As a consequence, it provides a strong protection against membership inference attacks. However, the model has to sacrifice a lot in terms of sample quality and diversity of synthetic images, making them not particularly useful for practical applications. For the specific task of generating class specific images, [23] introduces a differentially private extension to conditional GAN [24]. This can improve the downstream utility for a limited set of classification tasks. A comprehensive survey of differentially private GANs can be found in [25].

In addition to differential privacy based strategies, using techniques that improve generalization can also provide membership privacy benefits to generative models. Examples of such techniques are the use of dropout layers [26] and Wasserstein loss [27]. Unlike these techniques, the privGAN architecture is explicitly designed to maximize the privacy/utility trade-off, which is quantitatively demonstrated in Section 6.

3 priv(ate)GANs

In this section, we will motivate and introduce privGANs. In addition, we will provide the theoretical results for the optimal generator and discriminator.

3.1 The non–private GAN

Before introducing privGANs, let us define the original GAN. In the original (non–private) GAN, the discriminator D and the generator G play a two-player game to minimaximize the value function $V_0(G, D)$:

$$V_0(G, D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{1}$$

where p_z is the pre-defined input noise distribution, and p_{data} is the distribution of the real data X. Here, the goal of the generator is to learn realistic samples that can fool the discriminator, while the goal of the discriminator is to be able to tell generator generated samples from real ones. The solution to the minimax problem leads to a generator whose generated distribution is identical to the distribution of the training dataset [28].

3.2 Motivation of the privGAN architecture

Having defined the original GAN loss function, we note that one of its major privacy risks comes from the fact that the model tends to memorize the training samples [7, 15]. This leaves the original GAN vulnerable to some carefully constructed adversaries. For example, consider a situation where the trained GAN model has been open sourced, and a larger pool containing the training set is available to the public. [16] has constructed adversarial attacks that could easily identify the training samples using the observation that the trained discriminator is more likely to identify training samples as 'real' samples than those that were not in the training set. Similarly it has been shown in [18], that samples generated by trained generators are more similar to images in the training set than those not in the training set which can be utilized by an adversary to perform membership inference.

In the classification setting [8] demonstrated adversarial regularization as an effective strategy to prevent membership inference attacks. They used a built–in adversary to identify whether a sample was a part of the training set for the classification model or from a hold out set. The classification model eventually learns to make predictions that are ambigious enough to fool the adversary while trying to maintain the classification accuracy. Here, we adapt the same idea to generative models by introducing a built-in adversary which the generator must fool in addition to learning to generate realistic samples.

3.3 Membership inference attacks against GANs

To motivate the built–in adversary, let us first define a generic membership inference adversary for GANs. Given a training dataset X_{train}, a discriminator module D and a generator module G, the goal of a generic membership inference adversary is to learn the function $f(D, G, x)$ where:

$$Pr(x \in X_{train}) = f(D, G, x) \tag{2}$$

In the case of attacks that utilize only the discriminator model (e.g. the white–box attack in [16]), this reduces to the form:

$$Pr(x \in X_{train}) = f(D(x)) \tag{3}$$

Similarly, attacks that rely only on the generator model (e.g. the Monte–Carlo attacks described in [18]) reduce to the form:

$$Pr(x \in X_{train}) = f(G, x) \tag{4}$$

In the case of attacks on generators, all known attacks (to the best of our knowledge) rely on the distance of synthetic generated samples $G(z)$ with the sample of interest x (using some distance metric). The underlying assumption being that if a large number of synthetic samples have a small distance to x, then x was most likely a part of the training set.

The privGAN architecture is motivated by such an adversary. In addition, we demonstrate through empirical results that a protection against such adversaries also protects against adversaries that target discriminators. The privGAN architecture relies on two tricks to protect against such adversaries: i) random partitioning of the training dataset to train multiple generator–discriminator pairs, ii) a built–in adversary that must be fooled, whose goal is to infer which generator generated a synthetic sample. In the following sub–section we describe the privGAN loss highlighting these different parts.

3.4 Some notation relevant to privGAN

Before we introduce the mathematical formulation of privGAN, let us first list some important notation for ease of reading. Some of these terms will be defined in greater detail later on in the text:

- N denotes the number of generator-discriminator pairs in privGAN
- The tuple (G_i,D_i) denotes the ith generator-disciminator pair
- $X_1, \cdots, X_N$ denotes the partition of data corresponding to each generator-discriminator pair
- p_i refers to the distribution of the X_i
- p_z is some pre–defined input noise distribution
- D_p denotes the internal adversary or the 'privacy discriminator'
- λ is a hyperparameter that controls the privacy–utility tradeoff in privGAN
- $KL(p_a||p_b)$ stands for the KL-divergence between two distributions p_a and p_b.
- $JSD(p_1, \cdots, p_N)$ is the Jensen–Shannon divergence between the distributions $p_1, \cdots, p_N$.

3.5 The mathematical formulation of privGAN

Given an integer valued hyperparameter $N > 1$, we randomly divide the training data–set X into N equal sized non–overlapping subsets: $X_1, \cdots, X_N$. Each partition of the data is used to train separate generator–discriminator pairs (G_i,D_i) hence their cumulative loss is simply the summation of their individual value functions. We further introduce a built–in adversary (called the privacy discriminator) whose goal is to identify which generator generated the synthetic sample. In the case of $N = 2$, this is a similar objective to that of the adversary who only utilizes the generator model of a GAN. The loss of the privacy discriminator can then be written as:

$$R_p(D_p) = \mathbb{E}_{z\sim p_z(z)} \log[D_p^i(G_i(z))]$$

where, $D_p(x) = (D_p^1(x), \cdots, D_p^N(x))$ represents the probability of x to be generated by the generator G_i satisfying that $\sum_{i=1}^{N} D_p^i(x) = 1$. Hence, the complete value function $V_\lambda(\{G_i\}_{i=1}^N, \{D_i\}_{i=1}^N, D_p)$ for a privGAN is defined as:

$$\sum_{i=1}^{N} \big\{\mathbb{E}_{x\sim p_i(x)}[\log(D_i(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_i(G_i(z)))] \\ +\lambda\mathbb{E}_{z\sim p_z(z)} \log[D_p^i(G_i(z))]\big\},$$

where the p_i is the real data distribution of X_i for $i = 1, \cdots, N$, p_z is the pre-defined input noise distribution, $\lambda > 0$ is a hyperparameter that controls the privacy/utility trade–off. Figure 1A shows an illustration of the privGAN architecture when $N = 2$. It is easy to see that the complete value function takes the form :

$$\sum_{i=1}^{N} \underbrace{V_0(G_i, D_i)}_{utility} + \lambda \underbrace{R_p(D_p)}_{privacy}$$

Here, the first term optimizes for 'utility' (sample quality compared to training data partition) whereas the second term optimizes for membership privacy. Accordingly, the optimization problem for privGANs is

$$\min_{\{G_i\}_{i=1}^N} \max_{D_p} \max_{\{D_i\}_{i=1}^N} V_\lambda(\{G_i\}_{i=1}^N, \{D_i\}_{i=1}^N, D_p). \qquad (5)$$

In section 3.6 we will obtain the optimal solution to the above stated optimization problem. This result will then be used in section 3.7 to show that the the built–in adversary acts as a regularizer which prevents over–fitting to the training set partitions.

3.6 Theoretical results for privGANs

We first provide explicit expressions for the optimal discriminators given the generators.

Theorem 1. *Fixing the generators $\{G_i\}_{i=1}^N$ and the hyperparameter $\lambda > 0$, the optimal discriminators of Equa-*

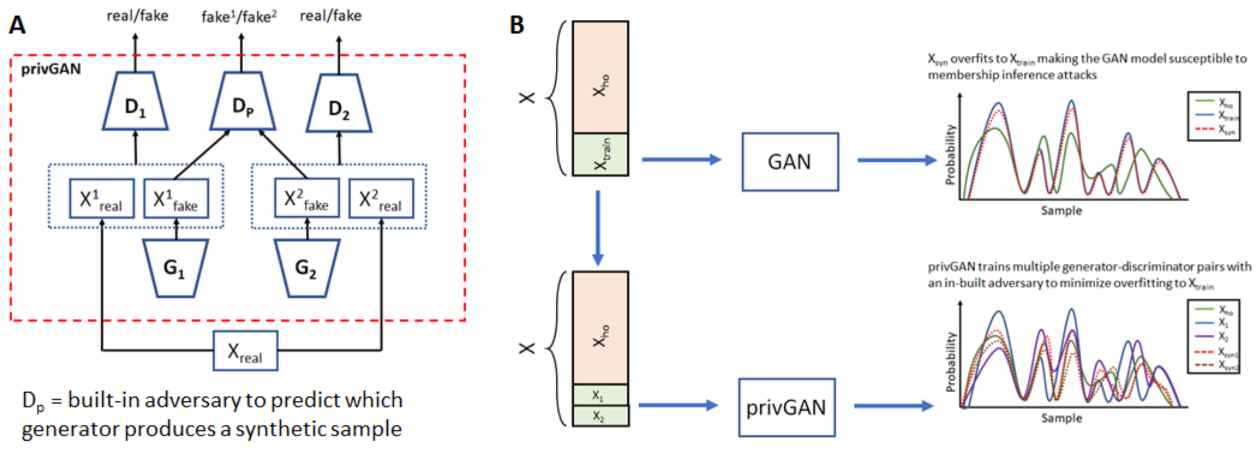
A
real/fake
fake¹/fake²
real/fake
privGAN
D1
Dp
D2
X¹real
X¹fake
X²fake
X²real
G1
G2
Xreal
Dp = built-in adversary to predict which generator produces a synthetic sample
B
X
Xho
Xtrain
GAN
Xsyn overfits to Xtrain making the GAN model susceptible to membership inference attacks
Probability
Sample
Xho
Xtrain
Xsyn
X
Xho
X1
X2
privGAN
privGAN trains multiple generator-discriminator pairs with an in-built adversary to minimize overfitting to Xtrain
Probability
Sample
Xho
X1
X2
Xsyn1
Xsyn2

Fig. 1. A) The privGAN architecture with 2 generator-discriminator pairs. B) Illustration of how privGAN provides protection against membership inference attacks by preventing memorization of the training set.

tion (5) are given by:

$$D_i^*(x) = \frac{p_i(x)}{p_i(x) + p_{g_i}(x)},$$
$$(D_p^i)^*(x) = \frac{p_{g_i}(x)}{\sum\limits_{j=1}^{N} p_{g_j}(x)}$$

for $i = 1, \cdots, N$, where p_{g_j} is the distribution of $G_j(z)$ given $z \sim p_z$ for $j = 1, \cdots, N$.

Proof. Decompose the value function as

$$V_\lambda(\{G_i\}_{i=1}^N, \{D_i\}_{i=1}^N, D_p) =$$
$$\sum_{i=1}^{N} V_0(G_i, D_i) + \lambda \sum_{i=1}^{N} \mathbb{E}_{z\sim p_z(z)} \log[D_p^i(G_i(z))],$$

where V_0 is defined in Equation (1). Note that the first term $\sum\limits_{i=1}^{N} V_0(G_i, D_i)$ only depends on $\{D_i\}_{i=1}^N$, while the second term $\lambda\mathbb{E}_{z\sim p_z(z)} \log[D_p^i(G_i(z))]$ depends on D_p alone. By Proposition 1 [28], $D_i^*(x) = \frac{p_i(x)}{p_i(x)+p_{g_i}(x)}$ maximizes $V_0(G_i, D_i)$ for $i = 1, \cdots, N$.

Note that (following [28])

$$\sum_{i=1}^{N} \mathbb{E}_{z\sim p_z(z)} \log D_p^i(G_i(z)) = \sum_{i=1}^{N} \mathbb{E}_{x\sim p_{g_i}} \log[D_p^i(x)]$$
$$= \int_x \sum_{i=1}^{N-1} p_{g_i}(x) \log D_p^i(x) + p_{g_N}(x) \log[1 - \sum_{j=1}^{N-1} D_p^j(x)]dx.$$

Then it is equivalent to solve the optimization problem $\max_{\{y_i\}_{i=1}^{N-1}} L(y_1, \cdots, y_{N-1})$, where $L(y_1, \cdots, y_{N-1}) = \sum\limits_{i=1}^{N-1} a_i \log y_i + a_N \log[1 - \sum\limits_{j=1}^{N-1} y_j]$ under the constraints that $\sum\limits_{i=1}^{N-1} y_i \in (0, 1)$ and $y_i \in (0, 1)$ for $i = 1, \cdots, N-1$. It is reasonable to assume that $a_i \geq 0$, since the probability density function is always positive. Easy to verify that $L(y_1, \cdots, y_{N-1})$ is concave, given any positive a_is. Note that $y_i^* = \frac{a_i}{\sum_{j=1}^{N} a_j}$ for $i = 1, \cdots, N-1$ solves the set of differential equations $\{\frac{\partial L}{\partial y_j} = 0\}_{j=1,\cdots,N-1}$ for any positive a_is. Thus it always maximizes $L(y_1, \cdots, y_{N-1})$ for any positive a_is, and we complete the proof. □

Similar to the original GAN [28], define

$$C_\lambda(\{G_i\}_{i=1}^N) = \max_{D_p} \max_{\{D_i\}_{i=1}^N} V_\lambda(\{G_i\}_{i=1}^N, \{D_i\}_{i=1}^N, D_p)$$

Theorem 2. *The minimum achievable value of $C_\lambda(\{G_i\}_{i=1}^N)$ is $-N(\log 4 + \lambda \log N)$ for any positive λ. This value is achieved if and only if $p_{g_i} = p_i = p_{data}$, for $i = 1, \cdots, N$.*

Proof. It is easy to verify that when $p_{g_i} = p_i = p_{data}$ for $i = 1, \cdots, N$, $C_\lambda(\{G_i\}_{i=1}^N)$ achieves $-N(\log 4 + \lambda \log N)$.

By its definition, $C_\lambda(\{G_i\}_{i=1}^N)$ can be re-written as:

$$C_\lambda(\{G_i\}_{i=1}^N) = \sum_{i=1}^{N} \mathbb{E}_{x\sim p_i}[\log D_i^*(x)]+$$
$$\mathbb{E}_{x\sim p_{g_i}}[\log(1 - D_i^*(x))] + \lambda\mathbb{E}_{x\sim p_{g_i}}[\log(D_p^{i*})(x)]$$

By Theorem 1 and a few algebraic manipulations, we have

$$C_\lambda(\{G_i\}_{i=1}^N) + N(\log 4 + \lambda \log N) = \sum_{i=1}^{N} \Big[KL(p_i || \frac{p_i + p_{g_i}}{2}) + KL(p_{g_i} || \frac{p_i + p_{g_i}}{2}) + \lambda KL(p_{g_i} || \frac{\sum_{j=1}^N p_{g_j}}{N}) \Big], \quad (6)$$

Note that the Jensen-Shannon divergence (JSD) between N distributions $p_1, ...p_N$ is defined as $\frac{1}{N} \sum_{i=1}^N KL(p_i || \frac{\sum_{j=1}^N p_j}{N})$. Then, Equation (6) turns out to be

$$\sum_{i=1}^{N} 2JSD(p_i || p_{g_i}) + N\lambda JSD(p_{g_1}, .., p_{g_N}) \geq 0,$$

where the minimum is achieved if and only if $p_{g_i} = p_i$, for $i = 1, \cdots, N$, and $p_{g_1} = \cdots = p_{g_N} = p_{data}$, according to the property of Jensen-Shannon divergence. Thus completing the proof.

□

Remark 1. *Assume that $p_i = p_{data}$ for $i = 1, \cdots, N$. Given $(\{G_i^*\}_{i=1}^N, \{D_i^*\}_{i=1}^N, D_p^*)$ - the optimal solution of Equation (5), Theorems 1 and 2 indicates that each pair in the set*

$$\left\{ (G_i, D_i)_{i=1,\cdots,N}, (\sum_{i=1}^N G_i/N, \sum_{i=1}^N D_i/N) \right\}$$

minimaximizes $V_0(G, D)$.

This remark suggests that privGANs and GANs yield the same solution, when the data distribution of each partition X_i is identical to that of the whole dataset X. This will be true, if there are infinite samples in each partition X_i, and p_i equals to the underlying distribution, where the training samples were drawn from.

3.7 privGAN loss as a regularization

In Theorem 2 and Remark 1, we have focused on the ideal situation where we could get access to the underlying distribution, where the training samples were drawn from. In such an ideal situation, there is no room for the membership inference attacks, thus a privGAN and a GAN yield the same solution. However, in a practical scenario, this is not true (due to unavailability of infinitely many samples), making white–box and black–box attacks against GANs effective. In the following lemma, we will demonstrate that the built–in adversary (D_p) serves as a regularizer that prevents the optimal generators (and hence the discriminators) from memorizing the training samples.

Lemma 1. *Assume that $\{(G_i^\lambda)^*\}_{i=1}^N$ minimizes $C_\lambda(\{G_i\}_{i=1}^N)$ for a fixed positive λ. Then minimizing $C_\lambda(\{G_i\}_{i=1}^N)$ is equivalent to*

$$\min_{\{G_i\}_{i=1}^N} \sum_{i=1}^N JSD(p_i || p_{g_i}) \quad \textit{subject to: } JSD(p_{g_1}, .., p_{g_N}) \leq \delta_\lambda, \quad (7)$$

where $p_{g_i} \sim G_i(z)$ given $z \sim p_z$ for $i = 1, .., N$, and $\delta_\lambda = JSD(p_{(g_1^\lambda)^*}, .., p_{(g_N^\lambda)^*})$.

Proof. Since λ and N are fixed, reformulate $\min C_\lambda(\{G_i\}_{i=1}^N)$ as

$$\min_{\{G_i\}_{i=1}^N} \sum_{i=1}^N JSD(p_i || p_{g_i}) + \frac{N\lambda}{2} JSD(p_{g_1}, .., p_{g_N}). \quad (8)$$

Assume that $\{(G_i^\lambda)^+\}_{i=1}^N$ solves Equation (7). It also minimizes $C_\lambda(\{G_i\}_{i=1}^N)$, since $\sum_{i=1}^N JSD(p_i || p_{(g_i^\lambda)^+}) + \frac{N\lambda}{2} JSD(p_{(g_1^\lambda)^+}, .., p_{(g_N^\lambda)^+}) \leq \sum_{i=1}^N JSD(p_i || p_{(g_i^\lambda)^*}) + \frac{N\lambda}{2} \delta_\lambda = C_\lambda(\{(G_i^\lambda)^*\}_{i=1}^N) = \min C_\lambda(\{G_i\}_{i=1}^N)$.

We will then show that $\{(G_i^\lambda)^*\}_{i=1}^N$ is a solution of Equation (7). If the above assumption is not true, then there exists $\{G_i\}_{i=1}^N$ such that $\sum_{i=1}^N JSD(p_i || p_{g_i}) < \sum_{i=1}^N JSD(p_i || p_{(g_i^\lambda)^*})$, and $JSD(p_{g_1}, .., p_{g_N}) \leq \delta_\lambda$. Then $\sum_{i=1}^N JSD(p_i || p_{g_i}) + \frac{N\lambda}{2} JSD(p_{g_1}, .., p_{g_N}) < \sum_{i=1}^N JSD(p_i || p_{(g_i^\lambda)^*}) + \frac{N\lambda}{2} \delta_\lambda = C_\lambda(\{(G_i^\lambda)^*\}_{i=1}^N) = \min C_\lambda(\{G_i\}_{i=1}^N)$. This contradicts the assumption that $\{(G_i^\lambda)^*\}_{i=1}^N$ minimizes $C_\lambda(\{G_i\}_{i=1}^N)$. This completes the proof. □

Theorem 2 and Lemma 1 provide an intuitive understanding of the properties of the optimal generator distributions. More specifically, it has been shown that the cost function $C_\lambda(\{G_i\}_{i=1}^N)$ reduces to a trade off between the distance of generator distributions and their corresponding data split, and their distance to the other generator distributions. On the one hand, the privacy discriminator can be seen as a regularization to prevent generators from memorizing their corresponding data split. On the other hand, a privGAN will yield the same solution as a non–private GAN when there is low risk of memorization, as suggested by Remark 1. This interesting observation further demonstrates the effectiveness of our protection, as the success of all kinds of membership inference attacks heavily rely on the extent of training set memorization. In addition, the reformulation of the optimization problem for the generators (seen in Lemma 1) provides a more explicit way to bound the distance between the generator distributions, which can be explored in future work to provide privacy guarantees. It should also be noted that the upper bound of $JSD(p_{g_1},..,p_{g_N})$ for $\{G_i^*\}_{i=1}^N$ will approach to 0 as $\lambda \to \infty$. According to the proof of Theorem 2, $C_\lambda(\{G_i\}_{i=1}^N)$ could be written as $-N\log 4 + \sum_{i=1}^N \left[KL(p_i||\frac{p_i+p_{g_i}}{2}) + KL(p_{g_i}||\frac{p_i+p_{g_i}}{2})\right] + N\lambda(JSD(p_{g_1},..,p_{g_N}) - \log N)$. When λ is large enough, $N\lambda(JSD(p_{g_1},..,p_{g_N}) - \log N)$ dominates the loss function. Note that $JSD(p_{g_1},..,p_{g_N}) \geq 0$, and it equals to 0 if and only if $p_{g_1} = .. = p_{g_N}$ almost everywhere. Hence, while λ and N are dataset dependent quantities, by increasing λ it should be possible to reduce $JSD(p_{g_1},..,p_{g_N})$ (for the optimal solution) to a user-desired value for most datasets.

Note: While the previous mathematical formulation and theoretical results are based on the original GAN formulation [28], the same idea of multiple generator-discriminator pairs and an internal adversary could be extended to most other GAN approaches such as WGAN [27],Conditional GAN [24] etc.. It must be noted though that the mathematical results will not automatically extend without modification to such cases.

3.8 Practical implementation of privGAN architecture

For the purposes of practical implementation of privGAN we just duplicated the discriminator and generator architectures of the simple GAN for all the component generator–discriminator pairs of privGAN. The privacy discriminator is identical in architecture to other discriminators barring the activation of the final layer, which is soft–max instead of sigmoid (in other discriminators). Identical learning rates are used as in the simple GAN. Hence, the only additional hyperparameters in privGAN are λ and the number of generator–discriminator pairs N. It should be noted that in the case of practical implementation, N lies in a bounded range $[2, \frac{|X_{train}|}{batchSize}]$ for a fixed batch size.

3.9 Training privGAN

The overall training algorithm for privGAN can be seen in Algorithm 1. While an alternating minimization strategy seems like a reasonable choice for training the privGAN, there are several practical tricks that can accelerate the convergence. The first trick is to set up good initial weights for the privacy discriminator (D_p). We first train a neural network to distinguish the different partitions of the training data (corresponding to each generator) for a small number of epochs n_i (here we used 50). Then initialize D_p with this neural network. The second trick is to fix D_p for the first n_d epochs (here we used 100) after the initialization, while only allowing the generator-discriminator pairs to train. These two strategies have been shown to accelerate the convergence. In addition, the learning curves for different combinations of n_i and n_d are presented in Figure 2. Setting $n_d = 0$ leads to big initial transients in the combined loss which eventually subsides. A possible explanation is that it is too hard for the generators to beat both discriminators and the privacy discriminator at the very beginning. The effect of setting $n_i = 0$ is less dramatic when $n_d = 100$. However, the convergence of the combined loss could be accelerated by setting $n_i = 50$, when $n_d = 0$. Note that D_p will be initialized with random weights by setting $n_i = 0$. Last but not least, the relative values of the various losses after 200 epochs are quite stable to the choice of n_i and n_d, as shown in Figure 2. Note - $n_i = 50$ and $n_d = 100$ are used in all experiments performed in the following sections.

4 Proposed attacks

In this section, we will introduce several state-of-the-art membership inference attacks against generative models from [16, 18], as well as an oracle attack of our own. Although the privGAN is specifically designed to defend against membership inference attacks targeting generators (or generated data), we also present other adversaries that attack released discriminator models for completeness.

4.1 White–box attack on discriminator

The white–box attack on a simple GAN is performed as outlined in [16]. Briefly, the attack assumes that the adversary is in possession of the trained model along with a large data pool including the training samples. The

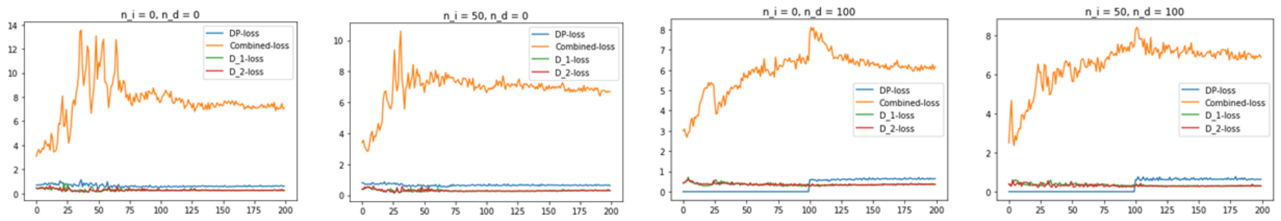

Fig. 2. Comparing the loss convergence for different model training hyperparameters of privGAN on the MNIST dataset with $\lambda = 1$.

Algorithm 1: Training privGAN

```
Input: Dataset $X$ with $m$ samples, array of generators $G$, array of discriminators $D$, privacy discriminator $D_p$, privacy weight $\lambda$, number of epochs $n$, initial $D_p$ epochs $n_i$, co-training delay $n_d$;
$N \leftarrow$ number of discriminators/generators;
Divide $X$ into $N$ equal parts $\{X_1, ..X_N\}$;
$y^p \leftarrow$ partition index of each data point;
for $i = 1$ to $n_i$ do
    Train one epoch of $D_p$ with $(X, y^p)$;
end
for $i = 1$ to $n$ do
    for $j = 1$ to $N$ do
        $X_j^f \leftarrow$ fake images generated with $G_j$;
        $y_j^f \leftarrow$ random numbers in $[1, N]$ excluding $j$;
        $y_j \leftarrow$ labels for fake and real images;
        $\hat{X}_j = \{X_j, X_j^f\}$;
        Train one epoch of $D_j$ with $(\hat{X}_j, y_j)$;
    end
    $X^f \leftarrow \{X_1^f, ..X_N^f\}$;
    $y^{fp} \leftarrow \{y_1^f, ..y_N^f\}$;
    if $i \geq n_d$ then
        Train one epoch of $D_p$ with $(X^f, y^p)$;
    end
    Train one epoch of $G$ with $(X^f, y^{fp}, y^p, \lambda)$;
end
```

attacker is also assumed to have the knowledge of what fraction of the dataset was used for training (say f) but no other information about the training set. The attack then proceeds by using the discriminator of the trained GAN to obtain a probability score for each sample in the dataset (see Algorithm 2). The samples are then sorted in descending order of probability score and the top f fraction of the scores are outputted as the likely training set. The evaluation of the white–box attack is done by calculating what fraction of the predicted training set was actually in the training set.

Algorithm 2: White-box attack on GAN

```
Input: Dataset $X$ with $m$ samples, discriminator $D$;
for $i = 1$ to $m$ do
    $p(X_i) \leftarrow D(X_i)$ ;
end
return $p$ ;
```

Since a privGAN model has multiple generator–discriminator pairs, the previously described attack cannot be directly applied to it. However, for a successful white–box attack, each of the discriminators should score samples from the training corpus higher than those not used in training (note: the training sets are of the same size for both private and non–private GANs). Hence, we modify the previous approach by identifying a *max* probability score by taking the max over the scores from all discriminators (see Algorithm 3). The rationale being that the discriminator which has trained on a particular data sample should have the largest discriminator score. We now proceed to sort the samples by each of these aggregate scores and select the top f fraction samples as the predicted training set, which is similar to Algorithm 3. We also tried taking *mean* instead of *max* which led to largely similar results, hence we only report the results for the *max* attack here.

Algorithm 3: White-box attack on privGAN

```
Input: Dataset $X$ with $m$ samples, array of discriminators $D$ ;
$N \leftarrow$ number of discriminators ;
for $i = 1$ to $m$ do
    $\{p^1(X_i), .., p^N(X_i)\} \leftarrow \{D_1(X_i), .., D_N(X_i)\}$ ;
end
$p^{max} \leftarrow max(p^1, .., p^N)$ ;
return $p^{max}$ ;
```

4.2 Oracle white–box attack on discriminator

While we describe a particular white–box attack targeting discriminator models from [16] in the previous sub–section, this is merely a heuristic based attack and there can be many other such heuristic attacks. A detailed analysis of all possible attacks is beyond the scope of this paper, but a taxonomy of possible attacks can be found in [17]. While the previously described white–box attack is an intuitive heuristic for a practical scenario, here we seek to identify the upper limit of membership inference accuracy of white–box attacks based solely on discriminator scores. This will enable us to better quantify the privacy loss due to GANs.

We first define the following notations:

- X_{train} is the training set for the GAN, and X_{ho} is the holdout set
- D is the discriminator
- $x \in X_{train} \cup X_{ho}$, is a sample which may or may not have been used to train the GAN.
- X is a random number drawn uniformly from $X_{train} \cup X_{ho}$
- $\mathcal{D} = \{D(x) | x \in X_{train} \cup X_{ho}\}$, and M is the size of $\mathcal{D}$. Sort the set $\mathcal{D}$ in the ascending order, and define d_i as its ith element for $i = 1, ..., M$.
- $\mathcal{A}(D(x)) \in \{0, 1\}$ is an adversary to infer the membership of a sample x using $D(x)$. More specifically, $\mathcal{A}(D(x)) = 1$ means that the adversary classifies the sample x as a training sample, otherwise it is classified into the holdout set X_{ho} ($\mathcal{A}(D(x)) = 0$).
- The utility score Δ of a membership inference attack $\mathcal{A}$ at a sample x is:

$$\Delta_{\mathcal{A}}(x) = \begin{cases} 1, & \text{if membership correctly identified} \\ -1, & \text{otherwise} \end{cases}$$

- $p_i = Pr(D(X) = d_i | X \in X_{train})$
- $q_i = Pr(D(X) = d_i | X \in X_{ho})$
- $f = Pr(X \in X_{train}) = \frac{|X_{train}|}{|X_{train}| + |X_{ho}|}$
- $s_i = Pr(D(X) = d_i) = p_i f + q_i(1 - f)$
- $\lambda_i = Pr(X \in X_{train} | D(X) = d_i) = \frac{p_i f}{p_i f + q_i(1-f)}$
- $\delta_i^{\mathcal{A}} = \mathcal{A}(D(d_i))$

Note that $\{p_i, q_i\}_{i=1,\cdots,M}$ could exactly describe the probability distribution of $D(X)$ given $X \in X_{train}$ or $X \in X_{ho}$. Thus, we are making no assumption about the distribution of $D(X)$. Next, we identify the maximum expected value of Δ for any adversary with access to p_i, q_i and f.

Theorem 3. *The maximum expected value of Δ is given by:*

$$\max_{\mathcal{A}} \mathbb{E}_X[\Delta_{\mathcal{A}}(X)] = \sum_{i=1}^{M} |p_i f - q_i(1 - f)|$$

The proof of Theorem 3 can be found in the Appendix.

It should be noted that such an oracle attack has a $\mathbb{E}_X[\Delta_{\mathcal{A}}(X)]_{max} = |2f - 1| \geq 0$ even when $P = Q$. We further show in Lemma 2 that for $f = \frac{1}{2}$, $\mathbb{E}_X[\Delta_{\mathcal{A}}(X)]_{max} = TVD(P, Q)$, where TVD stands for the Total Variation Distance between the distributions P and Q [29].

Lemma 2. *The maximum expected value of Δ is for $f = \frac{1}{2}$ is equal to the Total Variation Distance between probability distributions P and Q, where P is the distribution of discriminator scores given $x \in X_{test}$ while where Q is the distribution of discriminator scores given $x \in X_{ho}$.*

Proof. In the case of $f = \frac{1}{2}$, the condition $\lambda_i \geq \frac{1}{2}$ is equivalent to $p_i \geq q_i$. Hence, equation 15 can be re–written as:

$$\max \mathbb{E}_X[\Delta_{\mathcal{A}}(X)] = \frac{1}{2} \sum_{i=1}^{M} |p_i - q_i| \tag{9}$$

This is equal to the Total Variation Distance between probability distributions P and Q (Q.E.D.). □

Based on the previously stated observations, we use Total Variation Distance between the distribution of discriminator scores for X_{test} and X_{ho} to robustly quantify effectiveness of an attacker that only uses discriminator scores as described in Algorithm 4. Note - In the practical scenario, since the number of samples used in training are limited, we bin the discriminator scores into equally spaced bins and calculate the TVD on the resulting distributions.

Similar to the white–box attack described in the previous sub–section, this attack doesn't work directly

```
Algorithm 4: TVD attack on GAN
  Input: Datasets $X_{train}$ and $X_{ho}$ ,
  discriminator $D$, number of bins $M$ ;
  $p^i \leftarrow D(X_i), \forall i \in \{1, |X_{train}|\}$ ;
  $q^i \leftarrow D(X_i), \forall i \in \{1, |X_{ho}|\}$ ;
  $P \leftarrow$ probability distribution of $p^i$ with the
    range $[0, 1]$ discretized into $M$ equal bins ;
  $Q \leftarrow$ probability distribution of $q^i$ with the
    range $[0, 1]$ discretized into $M$ equal bins ;
  return $TVD(P, Q)$ ;
```

on privGAN due to the presence of multiple discriminators. Hence, we modify the attack by taking the maximum TVD of the distribution of discriminator scores for X_{test} and X_{ho} among all discriminators (Algorithm 5).

```
Algorithm 5: TVD attack on privGAN
  Input: Datasets $X_{train}$ and $X_{ho}$ , array of
  discriminators $D$, number of bins $M$ ;
  for $j = 1$ to $|D|$ do
      $p^{ij} \leftarrow D_j(X_i), \forall i \in \{1, |X_{train}|\}$ ;
      $P^j \leftarrow$ probability distribution of $p_{ij}$ with
        the range $[0, 1]$ discretized into $M$ equal
        bins ;
      $q^{ij} \leftarrow D_j(X_i), \forall i \in \{1, |X_{ho}|\}$ ;
      $Q^j \leftarrow$ probability distribution of $q_{ij}$ with
        the range $[0, 1]$ discretized into $M$ equal
        bins ;
  end
  return $\max_j(TVD(P^j, Q^j))$ ;
```

4.3 Monte–Carlo attack on generator

While the previous two sub–sections describe attacks on GANs that rely on using the discriminator scores to infer training set membership, it has been shown in [18] that generators of GANs are also vulnerable to membership inference attacks. [18] describes two attacks against generators, namely, instance membership inference and set membership inference. Instance membership inference is shown to work only marginally better than random chance guessing, while set membership inference was shown to be very effective in multiple datasets. Hence, in this paper we restrict ourselves to the set membership inference attack on generators only as described in [18].

In a set membership attack, we are given two sets of samples. All samples in one set have the desired membership, while all samples in the other set do not have said membership. The goal of the attacker is to identify which of the two sets contain samples with the desired membership. In the context of attacks against generative models, the goal is to identify the set that was part of the training set for the model.

```
Algorithm 6: Monte–Carlo set membership
attack on GAN/privGAN
  Input: Datasets $X_{train}$ and $X_{ho}$ , set size $m$,
  synthetic dataset $X$ of size $n$, distance metric $d$,
  epsilon $\epsilon$ ;
  $S_1 \leftarrow$, random subset of $X_{train}$ of size $m$ ;
  $S_0 \leftarrow$, random subset of $X_{ho}$ of size $m$ ;
  $y \leftarrow []$ ;
  for $j = 1$ to $m$ do
      $f_{S_1,\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{x_i \in U_{S_1,\epsilon}(x_i)}$ ;
      $f_{S_0,\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{x_i \in U_{S_0,\epsilon}(x_i)}$ ;
      if $f_{S_1,\epsilon} \geq f_{S_0,\epsilon}$ then
          $y[i] = 1$ ;
      else
          $y[i] = 0$ ;
      end
  end
  if $Sum(y) > \frac{m}{2}$ return 1 ;
  if $Sum(y) < \frac{m}{2}$ return 0 ;
  if $Sum(y) = \frac{m}{2}$ return $Bernoulli(0.5)$ ;
```

Algorithm 6 lays out the set membership inference algorithm described in [18]. The attack works by using the generator to generate some n number of samples (or using n generator generated samples if only samples are available). For each sample in the two sets whose memberships are being tested, we calculate $\frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{x_i \in U_{S_i,\epsilon}(x_i)}$, where $U_\epsilon(x) = \{x' | d(x, x') \leq \epsilon\}$. The authors find that the most effective distance is the a PCA based one. Where the top 40 principal components of the vectorized images in a held out set is first computed. To calculate the distance d between two images x and x', the PCA transformation is used to first compute the 40 principal components of interest. An euclidean distance is then computed between these reduced dimension vectors. The authors also prescribe several heuristics for choosing ϵ. The most effective heuristic is shown to be the median heuristic:

$$\epsilon = \underset{1\leq i\leq 2m}{\text{median}}(\min_{1\leq j\leq n} d(x_i, g_j))$$

Here g_j refers to the jth generated sample. We note that while [18] describes several minor variants of the same set membership attack (namely, different choices of distance metrics and heuristics for selecting ϵ), we choose the variant that was reported to have the best results in their experiments. We note that since all three of the datasets used in [18] are also used in our paper, this selection is well motivated.

5 Experiment details

5.1 Datasets used

We use the following standard open datasets for our experiments: i) MNIST, ii) fashion–MNIST, iii) CIFAR-10, and iv) Labeled Faces in Wild (LFW). MNIST and fashion–MNIST are grayscale datasets of size 70,000 (60,000 training samples, 10,000 test samples). MNIST comprises of images of handwritten digits, while fashion–MNIST contains images of simple clothing items. They contain a balanced number of samples from 10 classes of images. CIFAR-10 is a colored (RGB) dataset of everyday objects of size 60,000 (50,000 training samples, 10,000 test samples). LFW is a dataset of size 13,223 comprising of faces of individuals. We use the grayscale version of the dataset made available through scikit–learn.

5.2 Modeling and optimization

For MNIST, MNIST-fashion and LFW, we use standard fully connected networks for both generators and discriminators since these are relatively simple datasets. The generator and discriminator architecture details can be found in the Supplementary Methods. Identical generator and discriminator architectures are used for both GANs and privGANs. In the comparisons with DP-GANs we use an identical architecture as detailed in Supplementary Methods. While evaluating the different adversarial attacks, we trained all GAN models with an Adam [30] optimizer with a learning rate of 0.0002 ($\beta = 0.5$) for 500 epochs. While evaluating performance on downstream classification tasks, we train all GAN models with an Adam optimizer with a learning rate of 0.0002 ($\beta = 0.5$) for 200 epochs (except in CIFAR–10 where we train for 400 epochs as it is a much more complicated dataset). For the classifier, we use simple CNN models (see architecture in Supplementary Methods). For the CNN models, we still used a learning rate of 0.0002 but trained for 50 epochs instead since the model converges quickly. In all cases we used a batch–size of 256.

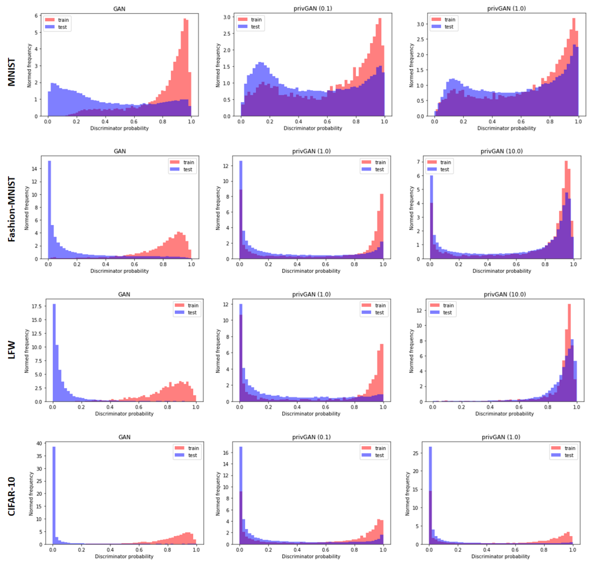

Fig. 3. Comparison of predicted scores by discriminators during white box attack on privGAN and non-private GANs for the various datasets. In the case of privGAN, the scores from one randomly selected discriminator. 'Train' refers to scores of data points in the training set, while 'test' refers to scores of data points not in the training set.

To test the efficiency of white–box and TVD attacks, models were trained on 10% of the data as in [16]. In the case of the Monte–Carlo attack, we first separated out the 'test set' for all datasets and used it only to compute the principal components as described in [18]. 100,000 synthetic samples (n) were used in the Monte–Carlo attack. 10% of the rest of the dataset was then used to train models while the model was evaluated on all the data except the held out test set. Reported numbers are averages over 10 runs. For each run, 10% of the dataset was randomly chosen to be the training set. In the case of the Monte–Carlo attack, 10 attacks were performed per run of model training and their average accuracy was taken (as in [18]). For the task of evaluating the downstream performance of GANs, a separate generative model was trained for each class of the training dataset. Here the training dataset refers to the pre–defined training set available for MNIST, MNIST–fashion and CIFAR–10.

6 Results

6.1 Comparison of privacy loss under the proposed attacks

A qualitative way to evaluate how well GANs are protected against white–box attacks is visually comparing the distribution of discriminator scores for samples in the training set with samples outside of the training set. The more similar the distributions are, the harder it is for an adversary to tell the samples apart. For a privGAN, since there are multiple discriminators, we can look at the outputs of a randomly chosen discriminator instead. In Figure 3 we see that the privGAN does indeed make the two distributions closer and the similarity between the distributions increases with λ. On the other hand, for a non–private GAN, the two distributions are very different which explains the high accuracy of white–box attacks in their case.

To quantitatively compare the privacy loss of privGANs with the baselines, we performed several attacks described in section 4. The first is a white–box attack as described previously. In the case of a white–box attack, since the privGAN has multiple generator/discriminator pairs, we describe a modified attack that is designed specifically for privGANs (see Algorithm 3). For each dataset, we train the GAN and privGANs (for $\lambda = 0.1, 1, 10$) on 10% of the dataset. The goal of the white–box attack is to then identify the training set from the complete dataset. Table 1 shows that increasing λ generally leads to reduction in the accuracy of white–box attacks. This indicates that a privGAN becomes more resistant to membership inference attacks, as λ becomes larger. Moreover, even for a small $\lambda = 0.1$, the privGAN leads to substantial decrease in accuracy of the white–box attack when compared to the non–private GAN for all datasets. In all cases, the privGAN model corresponding to the best performing value of λ yields comparable performance to the random chance. We also find that for two of the lower values of ϵ yielding usable images (25, 100), the white–box attack accuracy for DPGANs is similar to privGANs with $\lambda = 10$ (see Supplementary Table 7). To compare the effect of number of generators N on privacy, we also performed the white-box attack for varying N (2,4,6 and 8) with $\lambda = 1$ (see Supplementary Table 4). We see that increasing N generally leads to decrease in accuracy of the white-box attack, except in fashion-MNIST for $N = 8$. This may be either due to the heuristic nature of the attack or because unlike ϵ in differential privacy, the connection between λ or N to privacy is dataset dependent. We hypothesize that for certain datasets, increasing λ or N beyond certain optimal values, may cause decrease in sample quality or diversity, leading to lower membership privacy. For the MNIST and fashion-MNIST datasets, we also compare how white–box attack accuracy varies as a function of number of epochs for privGAN ($\lambda = 1$, $N = 2$) and non–private GAN (see Supplementary Figure 8). We find that, while increasing number of epochs increases the white-box attack accuracy for both privGAN and non-private GAN, privGAN performs significantly better than the non-private GAN at the same epoch number (against the white-box attack). It is therefore important that when comparing membership privacy benefits of privGAN against a non-private GAN, we train them for the same number of epochs. It is worth noting here that increase in privacy loss as a function of epochs is true for even differential privacy based techniques for a given level of input noise. However, unlike differentially private methods, which can bound the privacy loss for a given noise level and a given number of epochs [22], there is no direct way to do so for privGAN. In Section 7.3 we discuss potential data driven ways that may be used to estimate query specific membership privacy of models, which can then be used to study how the membership privacy of privGAN varies as a function of epochs.

Dataset	Rand.	GAN	privGAN		
			$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$
MNIST	**0.1**	**0.467**	**0.144**	**0.12**	**0.096**
f-MNIST	**0.1**	**0.527**	**0.192**	**0.192**	**0.095**
LFW	**0.1**	**0.724**	**0.148**	**0.107**	**0.086**
CIFAR-10	**0.1**	**0.723**	**0.568**	**0.424**	**0.154**

Table 1. White–box attack accuracy of various models on various datasets. For privGAN, the number represents accuracy of the 'max' attack.

While not a practical attack, the Total Variation Distance between the distribution of scores on the training and held out set provide an upper limit to the efficacy of discriminator score based white–box attacks against GANs (see Algorithm 4). It can be seen as an attack with an oracle adversary. Like in the previous case, this is complicated in the case of privGAN due to the presence of multiple generator–discriminator pairs. We mitigated this by taking the largest Total Variation Distance among all discriminators (see Algorithm 5. Similar to the previous attack, we trained the GAN and priv-

GANs (for $\lambda = 0.1, 1, 10$) on 10% of the dataset. We find again that for all datasets and all three values of λ, privGAN leads to considerable reduction in Total Variation Distance (Table 2). Moreover, an increase in λ is seen to generally lead to reduction in the Total Variation Distance. Here too we note that the reduction is not a monotonic as a function of λ, possibly for the reasons stated previously. Similar to the white-box attack, we also varied the number of generators keeping λ fixed (see Supplementary Table 5). We find that increasing N decreases the Total Variation Distance.

Dataset	GAN	privGAN		
		$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$
MNIST	0.438	0.31	0.235	0.048
f-MNIST	0.674	0.323	0.278	0.155
LFW	0.756	0.237	0.097	0.261
CIFAR-10	0.91	0.371	0.367	0.244

Table 2. TVD attack score of various models on various datasets.

While the previous two attacks are attacks against the discriminator, the final attack is one against the trained generators and only uses synthetic generated images. Moreover, unlike the previous two attacks which were instance membership inference attacks, this is a set membership inference attack (described previously). The attack (Algorithm 6), first described in [18], is a Monte–Carlo attack that tries to perform set membership inference under the assumption that generated images will be more similar to the image set used to train them. As seen in the previous two attacks, privGAN outperforms GAN for all three values of λ and set membership inference accuracy decreases as a function of λ (Table 3). Similar to the two previous attacks, we generally see a drop in attack accuracy as a function of N (see Supplementary Table 6), with the exception of fashion-MNIST for $N = 8$.

Dataset	Rand.	GAN	privGAN		
			$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$
MNIST	0.5	0.79	0.71	0.68	0.56
f-MNIST	0.5	0.75	0.73	0.7	0.64
LFW	0.5	0.77	0.66	0.57	0.55
CIFAR-10	0.5	0.62	0.61	0.56	0.52

Table 3. Monte–Carlo attack accuracy of various models on various datasets.

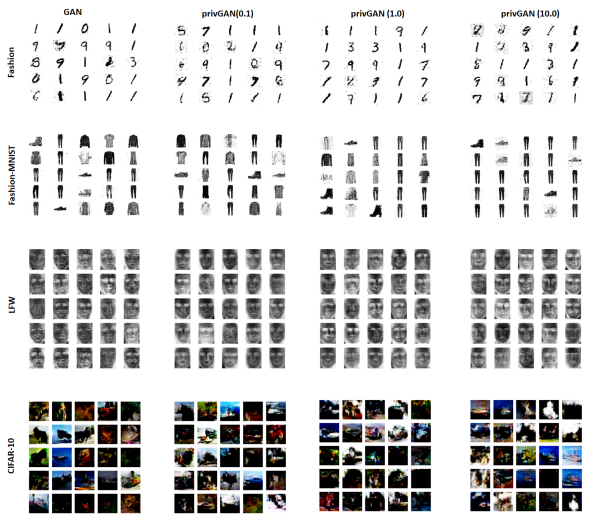

Fig. 4. Comparison of images generated by non-private GAN with privGAN for different values of λ. We see a gradual drop in quality of images with increasing values of λ.

6.2 Comparison of downstream performance against non-private GANs

We compare the downstream performance of privGANs against non–private GANs in two ways: i) qualitative comparison of the generated images, ii) quantitative comparison on a downstream classification task.

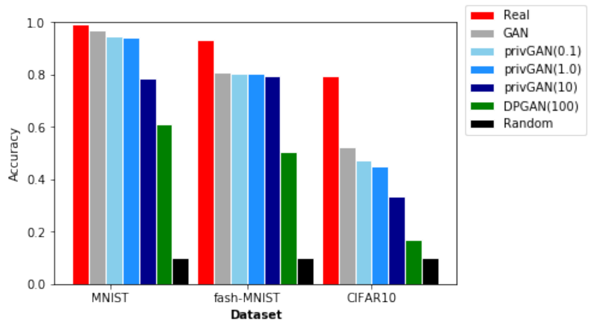

Fig. 5. Comparison of test–set performance of CNN models trained real data, synthetic data generated using GAN and synthetic data generated using privGAN. Numbers in brackets indicate λ values for privGAN and ϵ values for DPGAN.

For the first task we qualitatively compare the quality of images generated by privGANs with different settings of λ to those generated by non–private GANs as seen in Figure 4. It is easy to see that the image quality for all three λ values (0.1, 1, 10) are quite comparable to the images generated by non–private GANs. However, it can be seen that the image quality does decrease as we increase λ. We also see that certain classes become overrepresented as λ increases. This will be studied in greater detail in the following section.

To quantitatively test the downstream performance, we split the pre–defined training set for MNIST, MNIST–fashion and CIFAR by its class and trained privGANs ($\lambda = 0.1, 1, 10$) for each single class. We then generated the same number of samples per class as in the original training set to create a new synthetic training set(each image was generated by a randomly chosen generator). This training set was used to train a CNN classification model, which was then tested on the pre–defined test sets for each dataset. The baselines used for comparison were: i) CNN trained on the real training set, ii) CNN trained on a training set generated by a non–private GAN. Figure 5 shows that the classification accuracy decreases, as λ increases. However, the decrease is almost negligible with $\lambda = 0.1, 1$ compared to the non–private GAN for both MNIST and MNIST–fashion. Besides, a comparison with DPGAN (for 100) shows that privGAN leads to far higher downstream utility than DPGAN. While privGANs outperforms DPGANs for all three values of λ, we note that the comparison is not fair since DPGAN is known to provide a rather conservative privacy guarantees and isn't specifically designed with membership privacy in mind. Hence, a more fair comparison is between the privacy providing mechanism used in DPGAN (gradient clipping with gaussian noise addition) with that of privGAN. To do this, we do a white–box accuracy vs downstream accuracy plot for a wide range of λ (for privGAN) and ϵ (for DPGAN) values. (Note: This experiment was only performed on MNIST and fashion–MNIST due to computational considerations) We find that in both datasets, privGAN has higher utility for similar privacy for most of the privacy loss range as seen in Figure 6. It is also interesting to note that the privacy loss range for privGAN is somewhat smaller than DPGAN. This may indicate that simply sub-sampling the data already provides some amount of membership privacy. We tried to replicate this experiment for PATE-GAN [21] but were unable to generate reasonable samples, hence we don't report the results here. The implementation details and experiment results for PATE-GAN can be found in supplementary methods.

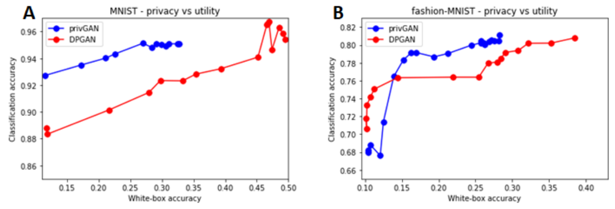

Fig. 6. Comparison of utility vs privacy between privGAN and DPGAN.

6.3 Effect of privGAN hyperparameters on sample quality

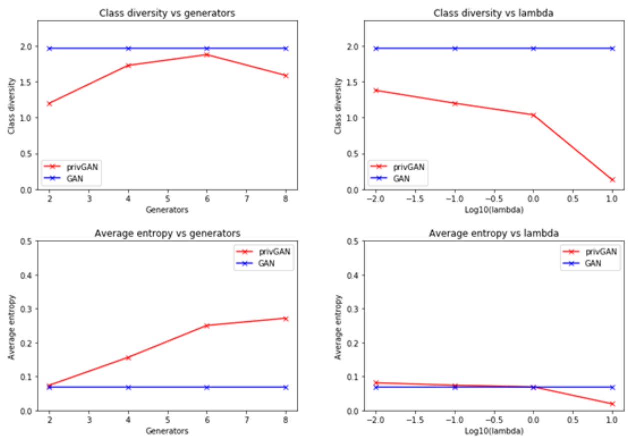

Fig. 7. Comparison of the effect of hyperparameters on average entropy and class diversity for MNIST.

To test the effect of the hyperparameter choices on sample quality we focus on two attributes: i) unambiguity of the class of the generated images, ii) relative abundance of different classes in generated images. We measure the unambiguity of the class of the generated images using the entropy of the predicted class probabilities (using a CNN trained on real images). The average entropy of the entire dataset is then reported for different hyperparameter settings (lower average entropy represents less ambiguity of image class). The class diversity of generated images is measured by using the pre–trained CNN to first identify the most probable class per sample and then using it to calculate the relative abundance of each class in the generated dataset (scaled to sum to 1). We then calculate the entropy of the relative class abundance which we report as the class diversity (higher entropy represents larger class diversity).

We see in Figure 7 that as λ (fixing number of generators to 2) is increased, both average entropy and class diversity monotonically decrease. This implies that as λ

increases, the class ambiguity of the samples increases, while the class diversity decreases. As the number of generators is increased (fixing $\lambda = 0.1$) we notice a monotonic increase of average entropy. This increase in average entropy is accompanied by an increase in class diversity (although the increase is not monotonic). This in turn implies that as the number of generators increases, the class ambiguity of samples decreases along with an increase in class diversity. Here it must be noted that as the number of generators is increased (for a fixed dataset size), the size of each data split decreases.

Based on these results, it can be summarized that both λ and the number of generators impact the quality of samples generated by privGAN. Since these two parameters interact, the optimal value of these hyperparameters are inter–dependent and most likely dependent on the dataset.

7 Practical considerations for data/model sharing

While the interplay between privacy and synthetic data sharing has been extensively discussed academically [31], there is a lack of discussion on when it is appropriate to share models (and synthetic data) and when it is recommended to share just synthetic data. These release choices can dramatically affect privacy concerns as different attacks against GANs can have vastly different success rates. In our case this is further complicated because privGAN has multiple generator discriminator pairs. Here we discuss some practical considerations for synthetic data/model sharing in the case of privGAN.

7.1 Sharing only synthetic data

Sharing only synthetic data is the most desirable option when possible, as it only allows for black box attacks, which are generally considered less effective than white box attacks for GANs [16]. While synthetic data sharing is relatively straightforward for non–private GANs, it is more complicated in the case of privGAN due to the presence of multiple generators, each of which may have lead to a slightly different generated distribution. Releasing images generated by any one generator may reduce the diversity of shared data, particularly for low values of λ. A smart sampling strategy (e.g. each image can be generated from a randomly sampled generator) from multiple generators would increase the diversity of the generated samples and would likely make it harder for adversaries to construct a black box attack. A relative privacy vs utility comparison of these approaches will be explored in future work.

7.2 Model sharing

While sharing only synthetic data should be generally more preferable to model sharing from a membership privacy standpoint, in many situations sharing the model might become necessary. Moreover, it is standard practice in many academic fields to share models for the sake of reproducible research. In such circumstances, it is strongly recommended to not share the D_p (privacy discriminator) module of privGAN. Since D_p serves the role of an in–built adversary in privGAN, it's ability to predict which generator generated a particular synthetic sample could be utilized by an adversary to infer membership. Additionally, one may consider releasing one or a randomly selected sub–set of generator–discriminator pairs instead of all pairs to further enhance membership privacy.

7.3 Certifying model/data privacy

While privGAN does not provide any privacy guarantees similar to differential privacy based techniques, some certificate of membership privacy can often be required by data/model owners as well as regulators. One could use the performance of realistic SOTA membership inference attacks against models or synthetic datasets as a way to empirically quantify membership privacy vulnerability of models. A more principled approach would be to use new privacy evaluation frameworks such as [32], that uses a Bayes Optimal Classifier as an optimal adversary and provides a dataset dependent and query specific membership privacy loss estimate (along with confidence intervals). This is a generalization of the Oracle attack introduced in this paper and is shown in [32] to be related to differential privacy. Although the optimal/Oracle adversary is an impractically strong adversary, the performance estimate of such an adversary can act as a post-hoc certificate for models/synthetic datasets. A data/model owner may decide to obtain such certificates for several different queries relevant to their application, to understand overall susceptibility of the model/synthetic data.

7.4 Hyperparameter choices

For practical deployment purposes, we suggest users first optimize hyperparameters for a non-private GAN (including number of epochs) and then set the privGAN hyperparameters λ and N. As evident from the experiment results, as well as the mathematical formulation, there is no dataset independent privacy interpretation for the hyperparameters λ and N. These quantities are strongly associated with the complexity and the size of datasets. Moreover, it matters what query function the MIAs can use to attack the model (e.g. discriminator scores in the case of model sharing or synthetic datasets). However, choice of these parameters affects both query specific membership privacy as well as downstream utility of the model/synthetic data. For the purposes of practical data/model sharing, we recommend performing grid search (or similar techniques) to identify optimal hyperparameters corresponding to query specific privacy (see above) and downstream utility (using the users choice of utility metric). These should allow users to identify privGAN hyperparameters that provide adequate privacy and utility for their application. A useful *heuristic* may be to first measure query specific membership privacy of a non-private GAN against a desired query and depending on the obtained value choose the range of λ or N to explore. In our experience, when MIAs are very effective against a non-private model, privGAN generally requires larger values of λ or N to obtain a desired level of membership attack accuracy. We should however note that increasing N has a larger impact on sample quality(Figure 7) than λ, and hence selecting very large values of N for small datasets may lead to poor model utility. We recommend starting with hyper-parameter values ($\lambda = 1, N = 2$) since this choice of hyperparameters has yielded reasonable results in the datasets we have tested on. Future work can be aimed at developing techniques for automatic identification of hyperparameters for a particular query and utility definition.

8 Conclusion

Here we present a novel GAN framework (named privGAN) that utilizes multiple generator-discriminator pairs and a built–in adversary to prevent the model from memorizing the training set. Through a theoretical analysis of the optimal generator/discriminators, we demonstrate that the results are identical to those of a non–private GAN. We also demonstrate in the more practical scenario where the training data is a sample of the entire dataset, the privGAN loss function is equivalent to a regularization to prevent memorization of the training set. The regularization provided by privGAN could also lead to an improved learning of the data distribution, which will be the focus of future work.

To demonstrate the utility of privGAN, we focus on the application of preventing membership inference attacks against GANs. We demonstrate empirically that while non–private GANs are highly vulnerable to such attacks, privGAN provides relative protection against such attacks. While we focus on a few state–of–the–art white–box and black–box attacks in this paper, we argue that due to the intrinsic regularization effect provided by privGAN (see Section 3.7) this would generalize to other attacks as well. We also demonstrate that compared to another popular defense against such attacks (DPGAN, PATE-GAN), privGAN minimally affects the quality of downstream samples as evidenced by the performance on downstream learning tasks such as classification. We also characterize the effect of different privGAN hyperparameters on sample quality, measured through two different metrics.

While the major focus of the current paper has been to characterize the properties of privGAN and empirically show the protection it provides to white–box attacks, future work could focus on theoretically quantifying the membership privacy benefits due to privGAN. Specifically, it would be useful to investigate connections between the privGAN hyperparameters (N and λ) and dataset memorization. The privGAN architecture could also have applications in related areas such as federated learning. Hence another direction of future work could be focused on extending privGAN to such application areas and demonstrating the benefits in practical datasets.

9 Acknowledgement

The authors acknowledge Dr. Ilya Mironov (Facebook), Dr. Jean-Francois Rajotte (University of British Columbia) and Mayana Pereira (Microsoft), for helpful discussions related to the paper. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *arXiv preprint arXiv:1802.06739*, 2018.

[2] H. A. Piwowar and T. J. Vision, "Data reuse and the open data citation advantage," *PeerJ*, vol. 1, p. e175, 2013.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, IEEE, 2017.

[4] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[5] L. Song, R. Shokri, and P. Mittal, "Membership inference attacks against adversarially robust deep learning models," in *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, IEEE, 2019.

[6] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Transactions on Services Computing*, 2019.

[7] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282, IEEE, 2018.

[8] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 634–646, 2018.

[9] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 259–274, 2019.

[10] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in supervised learning via generalization gap," *arXiv preprint arXiv:2002.12062*, 2020.

[11] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[12] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "Gan-based synthetic brain mr image generation," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 734–738, IEEE, 2018.

[13] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," *Medical image analysis*, p. 101552, 2019.

[14] R. Zheng, L. Liu, S. Zhang, C. Zheng, F. Bunyak, R. Xu, B. Li, and M. Sun, "Detection of exudates in fundus photographs with imbalanced learning using conditional generative adversarial network," *Biomedical optics express*, vol. 9, no. 10, pp. 4863–4878, 2018.

[15] K. S. Liu, B. Li, and J. Gao, "Generative model: Membership attack, generalization and diversity," *CoRR, abs/1805.09898*, 2018.

[16] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[17] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "Gan-leaks: A taxonomy of membership inference attacks against gans," *arXiv preprint arXiv:1909.03935*, 2019.

[18] B. Hilprecht, M. Härterich, and D. Bernau, "Monte carlo and reconstruction membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 232–249, 2019.

[19] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Revisiting membership inference under realistic assumptions," *arXiv preprint arXiv:2005.10881*, 2020.

[20] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

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

[22] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, ACM, 2016.

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

[24] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[25] L. Fan, "A survey of differentially private generative adversarial networks," in *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[26] W. Mou, Y. Zhou, J. Gao, and L. Wang, "Dropout training, data-dependent regularization, and generalization bounds," in *International Conference on Machine Learning*, pp. 3645–3653, 2018.

[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[29] R. M. Dudley, "Distances of probability measures and random variables," in *Selected Works of RM Dudley*, pp. 28–37, Springer, 2010.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] S. M. Bellovin, P. K. Dutta, and N. Reitinger, "Privacy and synthetic datasets," *Stan. Tech. L. Rev.*, vol. 22, p. 1, 2019.

[32] X. Liu, Y. Xu, S. Mukherjee, and J. L. Ferres, "Mace: A flexible framework for membership privacy estimation in generative models," *arXiv preprint arXiv:2009.05683*, 2020.

Appendix 1: Proof of theorem 3

Proof. The expected value of Δ is given by:

$$\mathbb{E}_X[\Delta_{\mathcal{A}}(X)] = 1 \times Pr(\text{correct prediction by } \mathcal{A}) - 1 \times Pr(\text{wrong prediction by } \mathcal{A}) \tag{10}$$

$$= \sum_{i=1}^{M}[\lambda_i\delta_i^{\mathcal{A}} + (1-\lambda_i)(1-\delta_i^{\mathcal{A}}) - (1-\lambda_i)\delta_i^{\mathcal{A}} - \lambda_i(1-\delta_i^{\mathcal{A}})]s_i \tag{11}$$

The optimal attack strategy is given by the following Integer Programming (IP) formulation:

$$\begin{aligned}\mathcal{A}^* = \arg\max_{\mathcal{A}} &\sum_{i=1}^{M}[\lambda_i\delta_i^{\mathcal{A}} + (1-\lambda_i)(1-\delta_i^{\mathcal{A}}) \\ &- (1-\lambda_i)\delta_i^{\mathcal{A}} - \lambda_i(1-\delta_i^{\mathcal{A}})]s_i \\ &\text{subject to:} \\ &\delta_i^{\mathcal{A}} \in \{0,1\}, \forall i \in \{1,..M\}\end{aligned} \tag{12}$$

This can be simplified to the following IP formulation:

$$\begin{aligned}\{(\delta_i^{\mathcal{A}})^\star\}_{i=1}^M = \arg &\max_{\{\delta_i^{\mathcal{A}}\}_{i=1}^M} \sum_{i=1}^{M}(2\lambda_i - 1)\delta_i^{\mathcal{A}} s_i \\ &\text{subject to:} \\ &\delta_i^{\mathcal{A}} \in \{0,1\}, \forall i \in \{1,..M\}\end{aligned} \tag{13}$$

It is easy to see that this IP has an analytical solution given by:

$$\delta^{\mathcal{A}^\star}_i = \begin{cases} 1, & \text{if } \lambda_i \geq \frac{1}{2} \\ 0, & \lambda_i < \frac{1}{2} \end{cases}, \forall i \in \{1,..,M\} \tag{14}$$

We note that the condition $\lambda_i \geq \frac{1}{2}$ is equivalent to $p_i \geq \frac{1-f}{f}q_i$.

Using $\{(\delta_i^{\mathcal{A}})^\star\}_{i=1}^M$, we find the maximum value of $\mathbb{E}_X[\Delta_{\mathcal{A}}(X)]$ to be:

$$\begin{aligned}\max \mathbb{E}_X[\Delta_{\mathcal{A}}(X)] &= \sum_{i=1}^{M}[\lambda_i \mathbf{1}_{p_i \geq \frac{1-f}{f}q_i} + (1-\lambda_i)\mathbf{1}_{q_i < \frac{1-f}{f}p_i} \\ &\quad - (1-\lambda_i)\mathbf{1}_{p_i \geq \frac{1-f}{f}q_i} - \lambda_i \mathbf{1}_{q_i < \frac{1-f}{f}p_i}]s_i \\ &= \sum_{i=1}^{M}[(2\lambda_i - 1)\mathbf{1}_{p_i \geq \frac{1-f}{f}q_i} + \\ &\quad (1-2\lambda_i)\mathbf{1}_{q_i < \frac{1-f}{f}p_i}]s_i \\ &= \sum_{i=1}^{M}|p_i f - q_i(1-f)|\end{aligned} \tag{15}$$

□

Appendix 2: Model architectures and hyperparameters

Here we outline the different layers used in the model architectures for different datasets, along with associated optimization hyperparameters. It is important to note that the same choices are made for non–private GAN, privGAN as well as DPGAN in all cases. Note that layers are in sequential order.

MNIST & MNIST–fashion

Generator layers

- Dense(units= 512, input size= 100)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 512)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 1024)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 784, activation = 'tanh')

Discriminator layers

- Dense(units= 2048)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 512)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 256)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 1, activation = 'sigmoid')

Privacy–Discriminator layers

- Dense(units= 2048)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 512)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 256)
- LeakyReLU($\alpha = 0.2$)
- Dense(units = number of generators, activation = 'softmax')

An Adam optimizer with $\beta = 0.5$ and a learning rate of 0.0002 was used for optimization.

LFW

Generator layers

- Dense(units= 512, input size= 100)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 512)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 1024)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 2914, activation = 'tanh')

Discriminator layers

- Dense(units= 2048)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 512)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 256)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 1, activation = 'sigmoid')

Privacy–Discriminator layers

- Dense(units= 2048)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 512)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 256)
- LeakyReLU($\alpha = 0.2$)
- Dense(units = number of generators, activation = 'softmax')

An Adam optimizer with $\beta = 0.5$ and a learning rate of 0.0002 was used for optimization.

CIFAR–10

Generator layers

- Dense(units= 2048, input size= 100, target shape= $(2, 2, 512)$)
- Conv2DTranspose(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2DTranspose(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2DTranspose(filters= 64, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2DTranspose(filters= 3, kernel size= 5, strides= 2, activation = 'tanh')

Discriminator layers

- Conv2D(filters= 64, kernel size= 5, strides= 2)
- Reshape(target shape= $(2, 2, 512)$)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2D(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Dense(units= 1, activation = 'sigmoid')

Privacy–Discriminator layers

- Conv2D(filters= 64, kernel size= 5, strides= 2)
- Reshape(target shape= $(2, 2, 512)$)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2D(filters= 128, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Conv2D(filters= 256, kernel size= 5, strides= 2)
- LeakyReLU($\alpha = 0.2$)
- Dense(units = number of generators, activation = 'softmax')

An Adam optimizer with $\beta = 0.5$ and a learning rate of 0.0002 was used for optimization.

CNN classifier for MNIST & MNIST–fashion

- Conv2D(filters= 32, kernel size= 3, activation = 'relu')
- Conv2D(filters= 32, kernel size= 3, activation = 'relu')
- Max–pooling(pool size= 2)
- Dense(units= 128, activation = 'relu')
- Dense(units= 10, activation = 'soft–max')

An Adam optimizer with $\beta = 0.5$ and a learning rate of 0.0002 was used for optimization.

CNN classifier for CIFAR–10

- Conv2D(filters= 32, kernel size= 3, activation = ’relu’)
- Conv2D(filters= 32, kernel size= 3, activation = ’relu’)
- Max–pooling(pool size= 2)
- Dropout(0.25)
- Conv2D(filters= 64, kernel size= 3, activation = ’relu’)
- Conv2D(filters= 64, kernel size= 3, activation = ’relu’)
- Max–pooling(pool size= 2)
- Dropout(0.25)
- Dense(units= 512, activation = ’relu’)
- Dropout(0.5)
- Dense(units= 10, activation = ’soft–max’)

An Adam optimizer with $\beta = 0.5$ and a learning rate of 0.0002 was used for optimization.

DPGAN hypterparameters and implementation

To make the architectures identical, we replaced the Wasserstein loss in DPGAN with the original GAN loss. In the implementation of DPGAN, there are several additional hyperparameters. In all our experiments, we have set $\delta = 1/N$, where N is the sample size of the dataset (this is considered standard practice). Furthermore, for each discriminator iteration, we performed 1 iteration of the generator for the sake of consistency with GAN and privGAN. DPGAN was implemented using the Tensorflow Privacy package (https://github.com/tensorflow/privacy).

PATE-GAN implementation

The generator architecture for PATE-GAN was chosen to be identical to a simple GAN. The Teacher & Student discriminators were chosen to have the same architecture as a simple GAN discrminator. The number of teacher discrminators was set to 5, following the default values found in the official bitbucket repository of project. The number of inner student and teacher discriminator iterations (n_S and n_T) were set to 5, also following default settings. The total number of iterations was chosen to identical to other types of GANs (*privGAN*, *DPGAN*) for different values of $\lambda \in [0.01, 50]$. The white–box attacks reported here were performed against the student discrminator of PATE-GAN. We note here that using these settings we were simply unable to generate any recognizable images using PATE-GAN. Furthermore, we were not able to find any examples in published/archived literature (or even blogs) where PATE-GAN has been used for image generation. On the contrary, others have noted that it is not suitable for image data generation [25].

Appendix 3: Effects of hyperparameter choices on membership privacy

Note: For experiments comparing different number of generators, LFW has been left out since the small size of the dataset leads to partition size being smaller than batch size for $n = 6, 8$. For experiments comparing different number of epochs, we restrict our comparison to MNIST and fashion-MNIST due to computational considerations.

Dataset	$n = 2$	$n = 4$	$n = 6$	$n = 8$
MNIST	0.12	0.095	0.098	0.08
f-MNIST	0.194	0.17	0.14	0.126
CIFAR-10	0.424	0.139	0.11	0.101

Table 4. White–box attack accuracy against privGAN for different number of generator/discriminator pairs. $\lambda = 1$ in all cases.

Dataset	$n = 2$	$n = 4$	$n = 6$	$n = 8$
MNIST	0.235	0.084	0.054	0.03
f-MNIST	0.278	0.235	0.166	0.101
CIFAR-10	0.367	0.247	0.138	0.056

Table 5. TVD attack score against privGAN for different number of generator/discriminator pairs. $\lambda = 1$ in all cases.

Dataset	$n = 2$	$n = 4$	$n = 6$	$n = 8$
MNIST	0.68	0.53	0.6	0.57
f-MNIST	0.7	0.69	0.5	0.6
CIFAR-10	0.56	0.55	0.56	0.52

Table 6. Monte–Carlo attack accuracy against privGAN for different number of generator/discriminator pairs. $\lambda = 1$ in all cases.

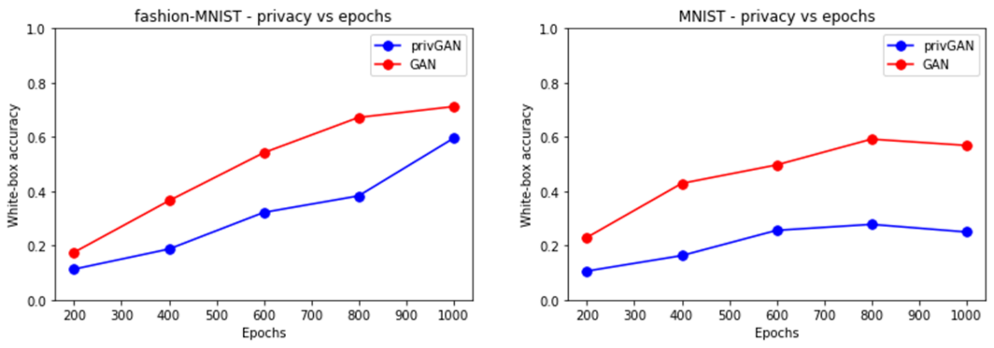

Fig. 8. Comparison of white–box privacy loss between privGAN and GAN for different number of training epochs (for fashion-MNIST and MNIST). $\lambda = 1$ and $N = 2$ for privGAN.

Appendix 4: Complete table of white–box attack efficiencies

Dataset	Rand.	GAN	privGAN ($\lambda = 0.1$)	privGAN ($\lambda = 1.0$)	privGAN ($\lambda = 10.0$)	DPGAN ($\epsilon = 100$)	DPGAN ($\epsilon = 25$)
MNIST	0.1	0.467	0.144	0.12	0.096	0.098	0.1
f-MNIST	0.1	0.527	0.192	0.192	0.095	0.102	0.099
LFW	0.1	0.724	0.148	0.107	0.086	0.109	0.097
CIFAR-10	0.1	0.723	0.568	0.424	0.154	0.107	0.098

Table 7. White box attack accuracy of various models on various datasets.

Dataset	GAN	privGAN ($\lambda = 0.1$)	privGAN ($\lambda = 1.0$)	privGAN ($\lambda = 10.0$)	DPGAN ($\epsilon = 100$)	DPGAN ($\epsilon = 25$)
MNIST	0.438	0.31	0.235	0.048	0.021	0.024
f-MNIST	0.674	0.323	0.278	0.155	0.022	0.025
LFW	0.756	0.237	0.097	0.261	0.04	0.045
CIFAR-10	0.91	0.371	0.367	0.244	0.01	0.008

Table 8. TVD attack score of various models on various datasets.

Dataset	Rand.	GAN	privGAN ($\lambda = 0.1$)	privGAN ($\lambda = 1.0$)	privGAN ($\lambda = 10.0$)	DPGAN ($\epsilon = 100$)	DPGAN ($\epsilon = 25$)
MNIST	0.5	0.79	0.71	0.68	0.56	0.62	0.6
f-MNIST	0.5	0.75	0.73	0.7	0.64	0.52	0.56
LFW	0.5	0.77	0.66	0.57	0.55	0.59	0.49
CIFAR-10	0.5	0.62	0.61	0.56	0.52	0.63	0.57

Table 9. Monte–Carlo attack accuracy of various models on various datasets.

Appendix 5: Privacy vs utility plots for PATE-GAN

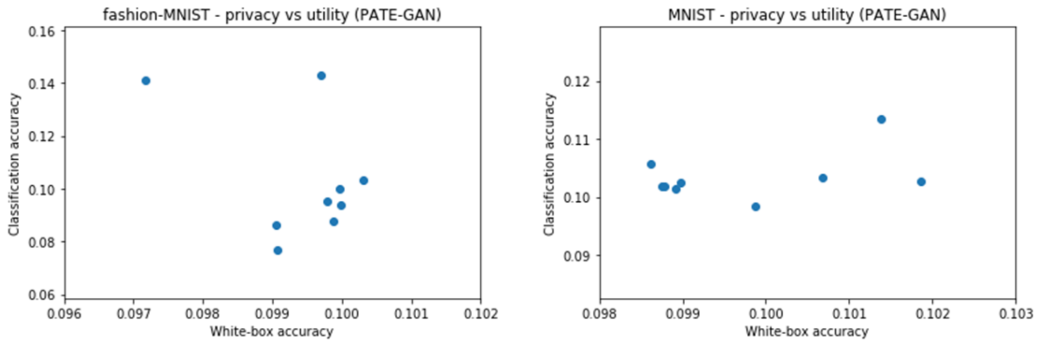

Fig. 9. Utility vs privacy for PATE–GAN on the fashiong-MNIST and MNIST datasets.