Privacy-Preserving Publication of Sensitive Data using Differentially Private GANs

Ricardo Silva Carvalho
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
ricardo_silva_carvalho@sfu.ca

Abstract

In order to publicly share sensitive data to support critical research or help solve some specific problem, generally we want to rigorously preserve the privacy of entries of the sensitive data while also still maintaining the usefulness of the released data. One promising solution for this setting is to use Generative Adversarial Networks (GANs) to learn the distribution of the sensitive data and generate synthetic data to be shared. However, previous work showed that using GANs does not preserves privacy, e.g. being susceptible to membership attacks. To address this issue we work with GANs within the Differential Privacy (DP) framework, which gives rigorous guarantees for the privacy of the sensitive data. More specifically, we build upon recent work on DP conditional GAN (DP-CGAN) describing a specific setting that generally improves utility of the results. Our source code was implemented from scratch on TensorFlow 2.0 and uses a custom made optimizer for DP-CGAN. Finally, we also demonstrate the usefulness of the DP-CGAN framework on a real world dataset of Thyroid disease, with empirical results showing just a negligible price to pay in terms of utility when adding privacy.

1 Introduction

Throughout the years, many research areas have been driven by publicly shared data. For example, several image datasets shared by the computer vision community are well-known and widely used for developing and testing new concepts. Despite the great value of these already public datasets, many important problems are still limited to certain research centers due to the inherent difficulties of sharing sensitive data. For example, we would increase the probability of achieving important discoveries in the health community if their sensitive datasets could become available to anyone interested in health problems. The concern is clear, sharing sensitive data can harm the individuals involved, and doing so while preserving the privacy is a challenge.

A first possibility to allow sharing sensitive datasets is to train a generative model that learns the data distribution, in order to allow further creation of synthetic data that resembles the original sensitive data while being fundamentally different. With this in mind, in this work we follow this idea focusing on the use of GANs: Generative Adversarial Networks [7]. GANs consist of two neural networks, named generator and discriminator. The generator creates synthetic data while trying to learn the sensitive data distribution by receiving noise as input and obtaining feedback from the discriminator, which takes the synthetic/fake data together with the sensitive/real data and learns how to classify input as real or fake. In practice, these two neural networks play a minimax game and are trained simultaneously.

Although a solution using GANs seems plausible, recent work already showed that the results obtained by GAN models are still vulnerable to attacks [14, 8]. For example, a GAN model can be

susceptible to membership inference attacks or model inversion attacks [6, 17]. Therefore, simply directly using GANs for synthetic data generation is problematic, as it still can expose information from the sensitive data.

Currently, one of the most widely adopted formal definitions for preserving privacy is Differential Privacy (DP) [4]. It ensures that the result of an algorithm satisfying DP will have very little impact if we remove or add any user from the dataset used to generate the result. Many companies have adopted differential privacy as a standard, such as Google [5], Microsoft [3] and LinkedIn [10].

More specifically, DP has also been used with machine learning. Intuitively, to use DP one first sets a bound on the maximum effect a row of the dataset can have on the output and then adds noise proportional to such bound. While training neural networks via stochastic gradient descent, the standard way of enforcing DP is to clip the L2 norm of each individual gradient and then add Gaussian noise to the clipped gradients.

A recently introduced framework using such method is DP-CGAN: Differentially Private Conditional GAN [15]. It enforces DP on an extension of the original GAN formulation, the Conditional GAN (CGAN) [12], where both generator and discriminator are conditioned on some additional information, such as class labels. DP-CGAN currently gives the best results on the MNIST dataset for a limited privacy budget and is one of the few works that made the source code available, which uses TensorFlow (TF) 1.15.

Our work builds upon DP-CGAN. First we implement DP-CGAN from scratch on TensorFlow 2.0, which is considerably different from the 1.15 version regarding the privacy package. To the best of our knowledge this is the first open source implementation of a differentially private GAN on TF 2.0. To enable the correct privacy computations of a differentially private GAN, we had to implement a new custom optimizer that adds noise to gradients only when needed. Additionally, after discussion with the main author of DP-CGAN and Google engineers that created the privacy package on TensorFlow, we realized that the original TF 1.15 code of DP-CGAN had inconsistencies, and we carefully avoid such problems when implementing our version on TF 2.0 to guarantee correctness.

Moreover, in this work we test three different learning strategies to verify if the utility of the DP-CGAN improves. To this end, we specifically consider that the DP-CGAN will be used to create a synthetic dataset to be shared with the goal of training a classification model on such dataset. In other words, the goal is to create a classification model for a sensitive dataset, but instead of training the model on the original sensitive data, we use a DP-CGAN to generate privacy-preserving synthetic data and then train the model on it. By defining this goal we can measure the usefulness of the DP-CGAN framework in terms of utility by measuring the quality of the model trained on the synthetic data. Therefore, in the end we want to assure privacy and get a reasonably good model as well. This benchmarking process is done on the MNIST dataset, and we show that one strategy, out of the three tested, results in improved AuROC (Area under ROC curve) of the model trained on the synthetic data when applied to the original testing data.

Finally, after testing these new strategies we then perform experiments with a medical dataset of patients with Thyroid disease. We illustrate the use of DP-CGAN for the case of an institution that wants to share its sensitive data while preserving privacy, in order to obtain a model to classify a given patient as normal, with hypothyroidism or hyperthyroidism. We show that even for scenarios of strict privacy the loss in utility is relatively small, giving empirical results in favor of using DP-CGAN in the suggested setting.

1.1 Related Work

To understand the guarantees provided in this work we start by defining differential privacy.

Definition 1 *A randomized mechanism $\mathcal{M}$ is (ε, δ)-differentially private if for any neighboring datasets D and D' that only differ by one row, and all outcome sets $\mathcal{O} \subseteq$ Range($\mathcal{M}$):*

$$\Pr[\mathcal{M}(D) \in \mathcal{O}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(D') \in \mathcal{O}] + \delta \tag{1}$$

The main idea for differential privacy is that the output distribution is roughly the same for datasets D and D' that differ only by one row. Therefore, guaranteeing that no single row will have a large impact on the output. From the definition we see that smaller values of ε and δ imply better privacy.

Now we describe important previous work focused on preserving privacy for GANs. In general, since the discriminator is the only neural network that looks at the sensitive data, it has the only learning process that needs to be differentially private. Additionally, we emphasize that clear privacy parameters ε and δ are needed to quantify privacy following Definition 1, therefore each DP learning process needs to specify a method to compute these parameters.

The first private GAN reported was DPGAN [16], which implements a differentially private version of WGAN [2] and adds Gaussian noise to the gradient of the Wasserstein distance for the discriminator. However, we note that the computation of the privacy parameters is not clearly defined and results on MNIST dataset still show considerable decrease in utility compared to non-private results.

Similarly, PATE-GAN [9] is another method that makes the discriminator differentially private. To achieve DP, PATE-GAN uses as discriminator a modified version of PATE [13] eliminating the need of public training data and using a teacher ensemble. Although PATE-GAN results are clear, the authors only evaluate it on small datasets and even though the architectures are complex, no source code is available. To track privacy parameters, PATE-GAN uses the Moments Accountant (MA) [1].

On the other hand, the DP-CGAN [15] framework, already described above, uses the recently introduced Renyi Differential Privacy (RDP) Accountant [11] to calculate the privacy parameters. In comparison to MA, RDP leverages better bounds which allow the addition of less noise while keeping the same privacy budget. Additionally, while DPGAN [16] adds noise to the gradients of the discriminator loss considering real and fake data together, DP-CGAN separately clips real and fake data, accumulates them and then add noise. This fact results in DP-CGAN adding less noise added overall, since only the real/sensitive data needs to be accounted for in terms of privacy concerns.

2 Approach

As mentioned before we build upon DC-GAN, testing different strategies during the learning process. Before we detail the algorithm we give a more complete overview of the parameters involved in training a differentially private model. Generally we have three privacy-specific hyperparameters:

- Clipping value (C): The maximum L2 norm of each gradient to be used for learning the model. This value bounds the effect each row in a dataset will have on the output.
- Noise scale (σ): The amount of noise to be considered when adding noise to the gradients during training. At each iteration, we add Gaussian noise with mean 0 and standard deviation σC. Generally, for other parameters fixed, increasing σ will result in better privacy.
- Number of microbatches (m): Each data batch of size B is split into m smaller units with B/m examples, called microbatches. Usually each microbatch contains one example, i.e. $m = B$, such that we clip gradients on a per-example basis. If we have $m < B$ we instead clip gradients after they have been averaged across the microbatch. Compared to $m = B$, having $m < B$ improves computational performance but decreases the utility of the result, as gradient averaging is done before adding noise, resulting in relatively more noise overall.

One important aspect to notice is that the original DP-CGAN paper did not mention the hyperparameter "number of microbatches" (m) and directly used $m = 1$, so changing this value will be one of our tested learning strategies. Below on Algorithm 1, we show DP-CGAN's description, however with a few changes from the description found on the original paper, namely:

- Now we do not compute privacy parameters at each iteration to define if we should keep going with the learning process. Instead we beforehand calculate the number of iterations we can run for given parameters using the RDP[1] accountant [11].
- We explicitly include the privacy hyperparameter "number of microbatches" m, which was inherently used by DP-CGAN but not specified or explained in the original paper. This leads to a more thorough gradient clipping description.

[1] Although to calculate the number of iterations we add the RDP function as **RDP**$(N, B, \sigma, \varepsilon, \delta)$, the actual process to find it is rather manual and is done iteratively, since the actual RDP function receives the number of iterations as input and outputs the ε. We just simplified the call for improved presentation.

- We specify that a separate random label sampling based on a given prior can also be defined, otherwise either the generator would query the real data and would then need to be differentially private like the discriminator, or the label distribution is considered known.

Algorithm 1 DP-CGAN

Input: Examples $\{x_1, x_2, ..., x_N\}$, labels $\{y_1, y_2, ..., y_N\}$, privacy parameters ε and δ, noise scale σ, clipping value C, learning rate η, batch size B, number of microbatches m
Output: Differentially Private Generator that creates synthetic data and labels: θ_g

$iterations$ = **RDP**$(N, B, \sigma, \varepsilon, \delta)$ ## Number of iterations to run

for $iter \in [iterations]$ **do**
Sample batch (X^t, Y^t) of size B with probability B/N from data distribution $p_{data}(X)$
Sample m microbatches (X_m^t, Y_m^t) of size B/m with probability $1/m$ from batch (X^t, Y^t)
Sample noise batch $Z^t = \{z_1, z_2, ..., z_B\}$ of size B from noise prior p_z
Sample label batch $L^t = \{l_1, l_2, ..., l_B\}$ of size B from label prior p_l or define $L^t = Y^t$

Compute discriminator loss on real data X^t and fake data Z^t
$disc\_loss\_real^t \leftarrow -\log(D(X^t, Y^t))$
$disc\_loss\_fake^t \leftarrow \log(1 - D(G(Z^t, L^t), L^t))$

Compute gradients of discriminator loss on real data X^t and clip them
for $X_m^t \in X^t$ **do** ## For each microbatch in a batch
for $x_i \in X_m^t$ **do** ## For each example in a microbatch
$disc\_grad\_real^t(x_i) \leftarrow \nabla_{\theta_d} disc\_loss\_real^t(\theta_d, x_i)$
end for
$disc\_grad\_real_m^t = \frac{m}{B} \sum_{x_i \in X_m^t} disc\_grad\_real^t(x_i)$
$disc\_grad\_real_m^t = disc\_grad\_real_m^t / \max(1, \frac{||disc\_grad\_real_m^t||_2}{C})$
end for

Compute gradients of discriminator loss on fake data Z^t and clip them
for $Z_m^t \in Z^t$ **do** ## For each microbatch in a batch
for $z_i \in Z_m^t$ **do** ## For each example in a microbatch
$disc\_grad\_fake^t(z_i) \leftarrow \nabla_{\theta_d} disc\_loss\_fake^t(\theta_d, z_i)$
end for
$disc\_grad\_fake_m^t = \frac{m}{B} \sum_{z_i \in Z_m^t} disc\_grad\_fake^t(z_i)$
$disc\_grad\_fake_m^t = disc\_grad\_fake_m^t / \max(1, \frac{||disc\_grad\_fake_m^t||_2}{C})$
end for

Compute overall gradients of discriminator and add Gaussian Noise to them
$disc\_grad^t \leftarrow \frac{1}{m} \sum_{i \in [m]} \left(disc\_grad\_real_m^t + disc\_grad\_fake_m^t + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Update discriminator
$\theta_d \leftarrow SGD(disc\_grad^t, \theta_d, \eta^t)$

Compute generator loss
$gen\_loss^t \leftarrow \log(1 - D(G(Z^t, L^t), L^t))$

Compute gradients of generator loss
$gen\_grad^t \leftarrow \nabla_{\theta_g} gen\_loss^t(\theta_g, Z^t)$

Update generator
$\theta_g \leftarrow ADAM(gen\_grad^t, \theta_g)$
end for
Return θ_g

3 Experiments

Here we perform two experiments: first we test new learning strategies to enhance utility of DP-CGAN, and then we apply it on real world data, illustrating the application for the health community.

3.1 Benchmarking

In this section we show how three different learning strategies behave in terms of results of AuROC. For each setting, first we train a DP-CGAN using a sensitive training dataset, then we create a synthetic dataset using the generator from DP-CGAN, after that we train a classification model on the synthetic dataset and finally we test the model created on the original testing dataset. For this benchmarking we use the MNIST dataset [2]. Complete source code is publicly available[3].

For fair comparison with original DP-CGAN, we use their exact same vanilla architecture, depicted on Figure 1, with fully connected layers containing respectively (from left to right) 128, 784, 128 and 1 nodes. The parameters are also the same the original DP-CGAN paper used: $B = 600$, $\sigma = 1.15$, $C = 1.1$, $\eta = 0.15$ to $\eta = 0.052$ in 10^3 iterations and constant $\eta = 0.052$ until reaching 249 epochs, $\varepsilon = 9.6$ and $\delta = 10^{-5}$.

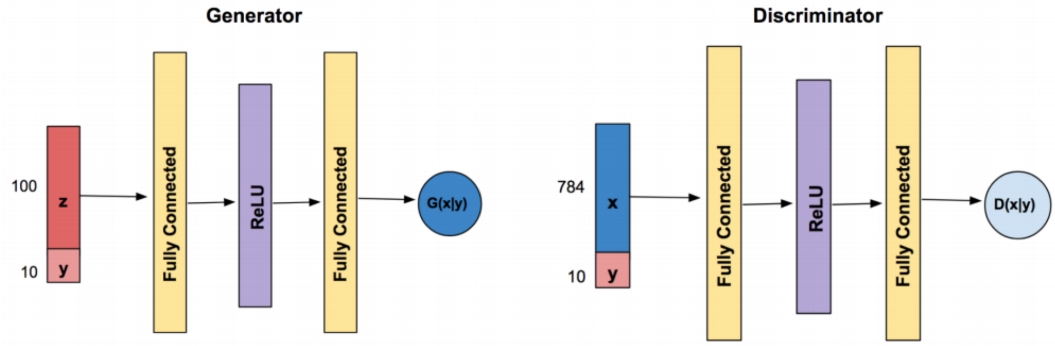

Figure 1: Vanilla CGAN Generator and Discriminator Architecture. Fully connected layers containing respectively (from left to right) 128, 784, 128 and 1 nodes. Image obtained from [15].

To train a classification model on the synthetic data we also use the same classifiers the original DP-CGAN paper used: standard Logistic Regression and Multilayer Perceptron models from sklearn[4] library without any grid search. We note that we are only keeping these settings to have a fair comparison, since those are the same settings of the original paper.

The strategies tested were:

1. Using number of microbatches m equal to batch size B. Original DP-CGAN used $m = 1$.
2. Strategy one plus updating discriminator for real and fake datasets separately. Original DP-CGAN updates them together.
3. Strategy one plus using LeakyRelU with $\alpha = 0.2$. Original DP-CGAN used ReLU.

Additionally, for baseline purposes we show the results for the case of training the classification model on the real training data and on a synthetic data generated from a non-DP CGAN.

From Table 1 we see that only the first strategy resulted in a significant improvement. Such improvement makes sense since, for $m = B$, when we look at Algorithm 1 we see that it does not average before adding noise and then "fully averages" after adding noise. Therefore, noise is smaller in proportion to signal. The only downside is that we clip each gradient per-example, thus having more computational effort.

[2]The MNIST dataset of handwritten digits: `http://yann.lecun.com/exdb/mnist/`.
[3]Source code for DP-CGAN on TF 2.0: `https://github.com/ricardocarvalhods/dpcgan`
[4]Python sklearn library: `https://scikit-learn.org/`.

Table 1: AuROC on testing data of MNIST using standard sklearn python library of models trained on synthetic/fake data. DP results are average of 3 trials.

AuROC	Real	CGAN	DP-CGAN TF 1.15	DP-CGAN TF 2.0		
			$m = 1$	$m = B$	+Sep. Bat.	+LeakyRelU
LR	0.9217	0.9110	0.8121	0.8642	0.6308	0.8088
MLP	0.9760	0.9106	0.8396	0.8858	0.6586	0.8263

Finally, although the other two strategies gave worse results, we note that they need further work, as in different settings (e.g. more complex architectures) they may result in improved AuROC.

3.2 Thyroid Disease Dataset

Now we demonstrate the use of DP-CGAN on the Thyroid disease dataset, containing information of 7200 patients, split into two training (52.4%) and testing (47.6%). It includes 21 attributes, with 15 being binary and 6 continuous. The patients are categorized into 3 classes: normal, hypothyroidism, hyperthyroidism. For this experiment, we used the following settings:

- Generator with same architecture as in Figure 1, but with second fully connected layer having only 21 nodes. Discriminator with exact same architecture as in Figure 1.
- Privacy parameters: $\varepsilon = 3.7$ and $\delta = 10^{-5}$ and DP-CGAN hyperparameters: $B = 32$, $m = B$, $\sigma = 1.15$, $C = 1.1$, $\eta = 0.15$ to $\eta = 0.052$ until reaching 50 epochs.
- Values of 10% above or below were manually tested for each DP-CGAN hyperparameter. To finally select the hyperparameters described above, on each run we used the cross validation result of the classification model trained on the synthetic data of the DP-CGAN as metric.
- Classification model was trained with Neural Network from sklearn (MLP) with just one hidden layer and fixed grid search (selecting settings with Cross Validation) containing hyperparameters: learning rate (10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}), optimizer (LBFGS, Adam) and hidden layer size (15, 16, 17, 18, 19, 20). Best model had learning rate of 10^{-8}, hidden layer size of 16, and LBFGS optimizer.

Results below include the AuROC for the model trained on the sensitive training data, in order to compare the loss in utility due to privacy preservation using DP-CGAN.

Table 2: AuROC on testing data for neural network model trained with real or fake data

	REAL	**DP-CGAN**
AuROC	0.9858	0.9746

From Table 2 we see that the loss in utility, represented by AuROC, is relatively small, of approximately 1%. This indicates a favorable result towards DP-CGAN, as we gain privacy guarantees and obtain a model from the synthetic data considerably similar to one that would be obtained from the original sensitive data.

4 Conclusion

In this work we showed how to create privacy-preserving data from a sensitive dataset using differentially private GANs. More specifically we have built upon DP-CGAN and showed a specific setting for learning making use of maximum number of microbatches, which improved the results compared to previous work. The DP-CGAN framework was then applied to real world dataset for Thyroid disease. Experimental results showed a negligible decrease in AuROC of the model trained on synthetic data generated from DP-CGAN compared to a model trained on the real sensitive data. Thus, this work shows promising initial results for continuing using GANs for privacy preserving data generation.

Acknowledgments

We thank the first author of the DP-CGAN [15], Reihaneh Torkzadehmahani, for clarifying the settings used in their work and for fruitful discussions about the framework. Additionally, we thank Google privacy engineers Galen Andrew, Steve Chien and Brendan McMahan for giving valuable insights and helping dissolve important questions about TF privacy.

References

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

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.

[4] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part II*, pages 1–12. Springer-Verlag, 2006.

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

[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019.

[9] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: generating synthetic data with differential privacy guarantees. 2018.

[10] Krishnaram Kenthapadi and Thanh TL Tran. Pripearl: A framework for privacy-preserving analytics and reporting at linkedin. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2183–2191. ACM, 2018.

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

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

[13] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

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

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

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

[17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.