Differentially Private Synthetic Data Generation via GANs

Extended Abstract[†]

Digvijay Boob[*]
Rachel Cummings[*]
Dhamma Kimpara[*]
Uthaipon (Tao) Tantipongpipat[*]
Chris Waites[*]
Kyle Zimmerman[*]

1 INTRODUCTION

Generating synthetic data is an attractive method for conducting private analysis of a sensitive dataset. It allows analysts to run their own non-private algorithms on the synthetic dataset without having to pre-specify the analyses they wish to perform. Further, both the dataset and any statistical results can be freely disseminated without incurring additional privacy loss. The goal of synthetic data generation is create data that will perform similarly to the original dataset for many analysis tasks.

In this working paper, we propose using a Differentially Private Generative Adversarial Network (DP-GAN) to generate private synthetic data. DP-GANs are a variant of Generative Adversarial Networks that are trained privately. GANs were first proposed by Goodfellow et al. [4], and there has since been a tremendous amount of research employing GANs to generate synthetic data. DP-GANs have recently been used for privately generating clinical trial data [2] and image datasets [8, 9]. We build off of previous work on DP-GANs and add further optimizations to enhance performance on wide variety of data types and analysis tasks. We also propose empirical validation of our algorithm's performance as future work.

2 BACKGROUND ON GANS

GANs are a type of generative model in which two neural networks, commonly known as the Generator (G_y) and Discriminator (D_w), are trained against each other in a zero-sum game. These neural networks are parameterized by their edge weights—y and w for G_y and D_w, respectively—which specify the function computed by each network.

The Generator takes as input a random vector drawn from a known distribution, and produces a new datapoint that (hopefully) has a similar distribution to the true data distribution. If we are given a finite-size database, then the true data distribution can be interpreted as the empirical distribution that would arise from sampling entries of the database with replacement. The Discriminator then tries to detect whether this new datapoint is from the Generator or from the true data distribution. If the Discriminator is too successful in distinguishing between the Generator's outputs and the true data, then this feedback is used to improve the Generator's data generation process.

We want to train D_w to maximize the probability of assigning right labels, whereas G_y should minimize the difference between its output distribution and true data distribution. The value of this two player zero-sum game between G_y and D_w can be written as following min-max optimization problem:

$$\min_y \max_w O(y,w) := \mathbb{E}_{x\sim p_{\text{data}}}[\log(D_w(x))]+\mathbb{E}_{z\sim p_z}[\log(1-D_w(G_y(z)))],$$

where p_{data} is true data distribution and p_z is a known noise distribution. In the min-max form of the game, D_w chooses w to maximize $O(y,w)$ and G_y chooses y to minimize $O(y,w)$. Their equilibrium strategies will achieve objective value $\min_y \max_w O(y,w)$. However, since $O(y,w)$ is a non-convex non-concave objective, these optimal strategies are typically not efficiently computable. Instead, we use gradient descent/ascent schemes to allow D_w and G_y to iteratively learn their optimal strategies.

We estimate the function and its gradients by sampling random elements from p_z and p_{data}. Let $\{z_1,\ldots,z_m\}$ and $\{x_1,\ldots,x_m\}$ respectively be random samples from p_z from p_{data}. We write $O_i(y,w) := \log(D_w(x_i)) + \log(1-D_w(G_y(z_i)))$ as i-th sampled function, and take the average value over the m samples to get estimate of O:

$$O_{\text{sample}}(y,w) = \frac{1}{m}\sum_{i=1}^{m} O_i(y,w).$$

Next we take the gradient with respect y and w: $g_y := \nabla_y O_{\text{sample}}(y,w)$ and $g_w := \nabla_w O_{\text{sample}}(y,w)$. Since the input data were randomly sampled, these are stochastic gradients. Finally, we do the gradient update step, with gradient ascent for D, $w \leftarrow w + \eta_w g_w$, and gradient descent for G, $y \leftarrow y - \eta_y g_y$, for step sizes η_w and η_y. This update process repeats either until the parameters converge or until a pre-specified number of update steps have occurred.

3 BACKGROUND ON DP-GANS

The original DP-GAN algorithm was proposed for private deep learning by Abadi et al. [1]. The algorithm privately trains the Discriminator because that neural network has access to the true data. The Generator only receives feedback about the true data through the Discriminator's output, and therefore will also be differentially private by post-processing.

In this algorithm, the Discriminator's stochastic gradient descent step is privatized by adding a gradient clipping and noise addition step. The algorithm clip gradient g_w to ensure that its norm is upper bounded by some constant C. We call this clipped gradient $g_{clip,w}$. This clipping ensures an upper bound of C on magnitude of the

[*]Georgia Institute of Technology. Email: {digvijaybb40, rachelc, dkimpara1, tao, cwaites3, zimmermankz}@gatech.edu. R.C. is corresponding author.

[†]Working paper based on preliminary results submitted to "The Unlinkable Data Challenge: Advancing Methods in Differential Privacy," posted by NIST.

gradient and hence the sensitivity of the update is at most C. To ensure differential privacy, the Gaussian Mechanism with variance σ is applied to $g_{clip,w}$ to get a noisy clipped gradient $g_{noise,clip,w}$. Finally, the parameter w is updated via gradient ascent using the noisy clipped gradient: $w \leftarrow w + \eta_w g_{noise,clip,w}$.

The algorithm of Abadi et al. [1] is differentially private, and makes use of a *moments accountant* to give even tighter privacy bounds than can be achieved through advanced composition. Suppose we choose $\sigma = \sqrt{2 \log \frac{1.25}{\delta}}/\epsilon$ then each update is (ϵ, δ)-differentially private. After T updates, advanced composition would give $(O(\epsilon\sqrt{Tk \log(1/\delta')}), Tk\delta + \delta')$-differential privacy. Using the moments accountant method, the overall algorithm is $(O(q\epsilon\sqrt{Tk}), \delta)$-differentially private for $q = m/n < 1$. Relative to advanced composition, this saves a factor of $\sqrt{\log(1/\delta')}$ in the ϵ-parameter and factor of Tk in the δ.

Theorem 3.1 ([1]). *There exist constants c_1 and c_2 so that given the sampling probability $q = m/n$ and the number of iterations T, for any $\epsilon < c_1 q^2 T$, Algorithm 1 is (ϵ, δ)-differentially private for any $\delta > 0$ if we choose*

$$\sigma \geq c_2 \frac{q\sqrt{T \log(1/\delta)}}{\epsilon}$$

In practice, this algorithm performs well. For example with $m = 0.01n, \sigma = 4, \delta = 10^{-5}$ and $T = 10000$, we have $\epsilon \approx 1.26$.

4 OPTIMIZATION TECHNIQUES TO IMPROVE ACCURACY

In this section, we summarize a number of techniques that can be used to improve the privacy-accuracy frontier for DP-GANs. These techniques are all combined with the DP-GAN framework in Algorithm 1.

4.1 Smart clipping

Different parameters in a neural network may have gradients of different scale, and hence ought to be clipped and injected with noise differently. This is particularly relevant for gradient coordinates that are small in magnitude, as adding relatively large amounts of noise to these coordinates may significantly harm accuracy. To address this, we use smart clipping techniques introduced in [9] to group gradient coordinates according to their relative magnitude, and add noise that only scales with the maximum magnitude in the group. Note that this is a grouping of parameters and not private data entries. Each group is clipped and the corresponding gradient is made appropriately noisy, so the update will remain private with respect to the true data. The grouping of parameters is determined dynamically in each inner loop of training by continuously maintaining the magnitude of each gradients, then performing clustering on those parameters.

We also adaptively choose the amount of clipping over time, which further improves accuracy and convergence rate. For each group of parameters in a given iteration, the amount of clipping is set to be the average of gradients of those parameters in the previous step.

4.2 Warm starting

Second, motivated by the observation that GANs tend to be unstable at the beginning of its training, previous work [9] has proposed to warm-start the DP-GAN by treating a small portion of the data ($\approx$ 2%) as public and use them to train the GAN non-privately. Zhang et al. [9] showed that this techniques improves accuracy by about 15% for standard statistical measures in machine learning. However, releasing even a small subset of sensitive data may raise legal or ethical concerns. Instead, we propose using publicly available data. As long as this public data is statistically similar to the sensitive data, it will provide the same accuracy improvements as subsampling. An analyst could similarly use domain knowledge or personal experience as a warm-start for DP-GAN.

4.3 Improved privacy via privacy accountant

The amount of noise added in each iteration is itself a random variable that depends on amount of gradient clipping and the grouping of parameters, which vary in each iteration based on previous random gradients. Therefore, directly applying advanced composition theorem on the general bound of noise added to gradients will result in a loose privacy bound. Intuitively, we should apply advanced composition theorem to the *actual* noise added to gradients. We keep track of our accumulated privacy loss using the privacy accountant of Abadi et al. [1], which relies on analysis of composing probabilities directly on the sequence of possible outcomes over iterations, rather than composing the sequence of privacy losses over iterations.

4.4 Task-specific optimization techniques

It is observed that, without further optimization, the loss function used for training DP-GANs converges (decreases) significantly at the start, but no longer consistently decreases in further training due to injected noise [2]. Instead, the loss varies within the range even after many epochs (outer iterations) of training. The problem is, then, how to choose the right parameters of DP-GAN from many epochs, when most of them perform within a range of accuracy? One method is to evaluate those parameters over many epochs with respect to a machine learning task of interest, then pick ones with top performance. We can still maintain privacy of the overall algorithm by using Report Noisy Argmax to pick the parameters.

We will first reserve some true data to privately train the appropriate model (e.g., random forest model for clustering) up to some satisfactory accuracy on the true data. We will then check the accuracy of this model on synthetic data generated from the generator at each outer iteration. Formally, let A denote the analysis task at hand, which takes in a dataset and outputs a classified. Let B be an accuracy evaluation task that takes in a classifier and labeled data (test data or holdout set) and output the accuracy of the classifier on the data. After training the DP-GAN for T epochs, we have a collection of T sets of DP-GAN generator parameters G^t for $t = 1, 2, \ldots, T$. For each $t = 1, 2, \ldots, T$, run algorithm A on synthetic dataset U_t generated by G^t to obtain a trained model θ_t. Then test the accuracy of θ_t on a holdout set of the the true data using algorithm B to obtain accuracy level α_t. Finally, we choose a small set $Q \subset [T]$ of the epochs by running Report Noisy Argmax on the set $\{\alpha_1, \ldots, \alpha_T\}$ without replacement. The final synthetic data S is generated by a combination of models: for each $t \in Q$, use G^t to generate a dataset S_t of $|S|/|Q|$ data points for desired size $|S|$ of the synthetic dataset. The final dataset is their union $S := \bigcup_{t \in Q} S_t$.

The algorithmic framework above applies to any analysis tasks including clustering and regression. For unlabelled data, such as for a clustering task, we can run the algorithm B on the holdout set and G^t, and measure the difference in performance with respect to some appropriate problem-specific metric. Alternatively, we can use statistical scores (e.g., Jensen-Shannon scores) which does not require any task to be specified, thus allowing for accuracy improvements in the task-independent setting as well. Then, we can proceed to choose Q by Report Noisy Argmax as usual.

4.5 GAN architecture for new data types

GANs have been recognized in machine learning for their ability to generate synthetic image data. In this section, we cite relevant existing work on the development of GANs for other types of data, and suggest our own ideas to utilize them in our DP-GAN.

Continuous data. Real-valued data is easily handled by neural networks and thus DP-GANs as well. See, for example, the success of using DP-GANs to generate clinical data, whose features are blood pressures of subjects over multiple visits [2].

Binary Data. It is folklore that neural networks are especially strong at classification tasks, relative to regression tasks. This is because classification can be thought of as a special and easier case of regression, where the output is restricted to specific set of numbers, e.g., $\{0, 1\}$ for binary-classification. For such a task, the output layer node generally has a perceptron activation which gives much clearer error signal for the back-propagation algorithm that calculates gradients. See [3] for a successful example of generating binary data in medical records.

Discrete-Valued Data. Motivated by the success of GANs in handling binary data, we propose encoding discrete data with small values (e.g., values below 15) as short binary strings, and treat discrete data with larger values as real-valued entries. We will round the synthetically generated data points to their nearest allowable integer value if they fall outside of the valid range.

Categorical Data. Categorical data can be handled with a newly-developed GAN architecture [5, 6] using the Gumbel-softmax function, which is a continuous approximation of multinomial distribution parameterized by a softmax function. We plan to incorporate this into our DP-GAN to handle categorical data.

Geo-spatial data. Geospatial data can be pre-processed by encoding geographical location as a two-dimensional real-valued attribute containing latitude and longitude. If the geospatial attribute describes a region (e.g., city or neighborhood), we can either randomly sample a point within that region or chose the center of the region. Once the attribute has been made numerical, we can use auxiliary information to post-process and improve accuracy. For example, if a randomly generated address appears in a body of water, that should be projected to a nearby location on land.

Graphs. GANs have recently been used to generate synthetic graphs by embedding the graph compactly into vectors specifically designed for GANs [7]. Our DP-GAN can also use this embedding techniques to privately generate synthetic graphs that share the same statistical properties as the original graph. This will allow private analysis of research questions such as link prediction, community detection, and influence maximization.

Algorithm 1 Minibatch SGD Algorithm For training DP-GANs

```
**Input parameters**: number of data samples in private training data $n$; public dataset $\mathcal{D}_{\text{public}}$; number of inner iterations $k$; learning rates $\eta_y, \eta_w$; minibatch size $m$; minibatch size for public data $m_{\text{public}}$; noise scale $\sigma$; number of parameter groups $l$; privacy budget $(\epsilon_0, \delta_0)$.
while y has not converged do
  for k steps do
    Run SGD-Batch($\mathcal{D}_{\text{public}}, \eta_w, m_{\text{public}}, l$)
  end for
  sample minibatch of $m$ noise samples $\{z_1, \dots, z_m\}$ from $p_z$
  compute stochastic gradient $g_y$
  update generator, $G_y$, by descending along stochastic gradient
      $y \leftarrow y - \eta_y g_y$
  query moments accountant with $\sigma, \epsilon_0, q(= m/n), k, T$ where
  $T$ is current number of outer iteration
      $\delta \leftarrow \exp\left\{ -\frac{\sigma^2 \epsilon_0^2}{c_2^2 q^2 T k} \right\}$
  if $\delta > \delta_0$ then
    break
  end if
end while
Note: Instead of using the standard gradient-based update rule, one can use momentum-based methods for gradient descent/ascent steps, which are faster and more convenient to use in practice.
```

Algorithm 2 SGD-Batch

```
**Input parameters**: public dataset $\mathcal{D}_{\text{public}}$; learning rate $\eta_w$; minibatch size for public data $m_{\text{public}}$; number of parameter groups $l$.
sample minibatch of $m_{\text{public}}$ noise samples $\{z_i\}_{i=1}^{m_{\text{public}}}$ from $p_z$
sample minibatch of $m_{\text{public}}$ data samples $\{\overline{x}_i\}_{i=1}^{m_{\text{public}}}$ from $\mathcal{D}_{\text{public}}$
compute corresponding gradient for each datapoint $\{\overline{g}_w^{(i)}\}_{i=1}^{m_{\text{public}}}$ where
    $\overline{g}_w^{(i)} := \nabla_w O_i(y, w)$
group parameter $w$ into $l$ groups: $\{G_j\}_{j=1}^{l}$, using weight clustering
    $\{(G_j, c_j)\}_{j=1}^{l} \leftarrow \text{weight-clustering}(l, \{\overline{g}_w^{(i)}\}_{i=1}^{m_{\text{public}}})$    (1)
sample minibatch of $m$ data samples $\{x_1, \dots, x_m\}$ from $p_{\text{data}}$
sample minibatch of $m$ noise samples $\{z_1, \dots, z_m\}$ from $p_z$
compute stochastic gradient $g_w$
clip gradient $g_w$ according to grouping $(G_j, c_j)$ obtained from weight clustering
    $g_{w,(j)} \leftarrow (g_w \cap G_j)$,    for $j = 1, \dots, l$    (2)
    $g_{w,(j)} \leftarrow g_{w,(j)} \min(1,\ c_j/\|g_{w,(j)}\|)$,    for $j = 1, \dots, l$    (3)
add corresponding noise in the clipped gradient groups
    $g_{w,(j)} \leftarrow g_{w,(j)} + \mathcal{N}(0, \sigma c_j \mathbf{I})$, for $j = 1, \dots, l$
update discriminator, $D_w$, by ascending along stochastic gradient
    $w_j \leftarrow w_j + \eta_w g_{w,(j)}$, for $j = 1, \dots, l$
    $w \leftarrow \{w_j\}_{j=1}^{l}$
```

5 PROPOSED EMPIRICAL EVALUATION

Our next step in this research agenda will be to implement and test the performance of the DP-GAN in Algorithm 1. The algorithm primarily combines existing techniques for improving the accuracy of DP-GANs. We will want to show that combining these techniques significantly improves accuracy over using any single technique alone.

Integrated Public Use Microdata Series (IPUMS) is one of the largest population databases available online, consisting of historical samples from both United States and international census records. Census extracts include a wide variety of selectable attributes, spanning numerical, categorical, and geospatial data types. Recently IPUMS put forth a preliminary data release of the 1940 United States full census extract consisting of approximately 130 million entries. This included demographic, economic, and location information on both the household and individual level, including attributes representing income, race, and census district. Given the recency of its release, it has been relatively unexplored by external regression, classification, and clustering analyses, leaving much room for novel investigation.

We believe this to be an interesting and ideal use case for several reasons. Firstly, GANs generally perform with higher accuracy as the number of samples increases. We hypothesize that with 130 million entries, the full census extract will be sufficiently large for our model to approximate the underlying distribution of data with high accuracy. Secondly, given that the census attributes span numerical, categorical, and geospatial data types, it represents a instance commonly faced in practice where data types vary, and a situation that our techniques are intended to address. Finally, given the U.S. Census Bureau's commitment to implementing differential privacy in the 2020 Census, this can serve as a testbed for private and accurate analysis of sensitive Census microdata.

REFERENCES

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

[2] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, and Casey S. Greene. 2017. Privacy-preserving generative deep neural networks support clinical data sharing. (2017). bioRxiv preprint 159756.

[3] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. 2017. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. In *Proceedings of Machine Learning for Healthcare Conference*. 286–305.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*. 2672–2680.

[5] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-softmax. (2016). arXiv preprint 1611.01144.

[6] Matt J Kusner and José Miguel Hernández-Lobato. 2016. GANs for sequences of discrete elements with the Gumbel-softmax distribution. (2016). arXiv preprint 1611.04051.

[7] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2017. Graphgan: Graph representation learning with generative adversarial nets. (2017). arXiv preprint 1711.08267.

[8] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. (2018). arXiv preprint 1802.06739.

[9] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially Private Releasing via Deep Generative Model. (2018). arXiv preprint 1801.01594.