Private Synthetic Data Generation via GANs (Supporting PDF)

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

August 2, 2018

Abstract

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.

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 have been trained privately, and can be used to generate an arbitrary amount of private synthetic data. 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.

Sections 1 and 2 provide background on GANs and DP-GANs, respectively. Sections 3 and 4 contain details of our proposed improvements over previous work. Section 5 contains our proposed algorithm.

1 Background on GANs

Generative Adversarial Networks (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. GANs were first proposed by [GPAM+14], and there has since been a tremendous amount of research employing GANs to generate synthetic data.

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

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

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 $O(y, w)$ 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.

An example GAN training algorithm is given below in Algorithm 1. Other GAN training algorithms may use momentum-based methods instead of the gradient-based update rule given here. Momentum-based methods often converge faster and are more convenient to use in practice.

Algorithm 1 Minibatch SGD Algorithm For training GANs

for number of training iterations **do**
  for k steps **do**
    sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ from p_z
    sample minibatch of m data samples $\{x_1, \ldots, x_m\}$ from p_{data}
    update discriminator, D_w, by ascending along stochastic gradient

$$w \leftarrow w + \eta_w g_w$$

  end for
  sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ from p_z
  update generator, G_y, by descending along stochastic gradient

$$y \leftarrow y - \eta_y g_y$$

end for

More broadly, GANs are a type of neural networks, which have been used to fit data using Stochastic Gradient Descent (SGD) [GPAM+14, ZBH+16]. Neural networks trained with SGD are already in use for variety of applications, including visual object classification [KSH12], natural language processing [CW08], and speech recognition [MDH12].

2 Background on DP-GANS

The first DP-GAN algorithm was proposed for private deep learning by [ACG+16], and is presented below in Algorithm 2. 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.

Algorithm 2 Minibatch SGD Algorithm For training DP-GANs

Input parameters: number of data samples n; number of inner iterations k; learning rates η_y, η_w; minibatch size m; noise scale σ; gradient bound C.

for number of training iterations **do**

for k steps **do**

sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ from p_z

sample minibatch of m data samples $\{x_1, \ldots, x_m\}$ from p_{data}

compute stochastic gradient g_w.

clip gradient g_w and then add noise.

$$g_w \leftarrow g_w \min(1, C/\|g_w\|) + \mathcal{N}(0, \sigma C \mathbf{I})$$

update discriminator, D_w, by ascending along stochastic gradient

$$w \leftarrow w + \eta_w g_w$$

end for

sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ from p_z

update generator, G_y, by descending along stochastic gradient

$$y \leftarrow y - \eta_y g_y$$

end for

Instead of using standard gradient based update rule, one can use momentum based methods which are faster and convenient to use in practice.

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 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: $w \leftarrow w + \eta_w g_{noise,clip,w}$.

Theorem 2.1 ([ACG+16]). *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 2 is (ϵ, δ)-differentially private for any $\delta > 0$ and for

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

The proof of this theorem 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 an (ϵ, δ)-differentially private instantiation of the Gaussian Mechanisms. 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. The factor of $q = m/n < 1$ arises from the additional privacy protection from sampling only $m \ll n$ data points in each update step for D_w. By privacy amplification, this makes each update step $(q\epsilon, q\delta)$-differentially private.

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

3 General 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 3.

3.1 Smart clipping

We first recall clipping gradient procedure described in answers to the questions and earlier section. We clip the gradient g_w during the training of DP-GAN to ensure that its norm is upper bounded by some constant C. We call this clipped gradient $g_{\text{clip},w}$. Then, to ensure differential privacy, we add $N(0, kC)$ noise (for input parameter k) to $g_{\text{clip},w}$, and refer to this noisy clipped gradient as $g_{\text{noise,clip},w}$.

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. Further, small gradients imply low sensitivity, so we should not need to inject as much noise to these coordinates to preserve same level of privacy

To address this, we use two smart clipping techniques introduced in [ZJW18]. The first technique is parameter grouping: we dynamically group different coordinates of gradients 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. Since each group is clipped and corresponding gradient is made appropriately noisy, the update will remain private with respect to private 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. The initial grouping can be determined from the gradients of first iteration, but the grouping obtained from pre-training the network with small public data demonstrates a more stable result. The pre-training idea, which we call *warm starting*, is discussed in Section 3.2.

The second smart clipping technique is adaptive clipping, which adaptively choose the amount of clipping over time. 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. This was shown in [ZJW18] to further improve the accuracy and convergence rate of DP-GANs.

3.2 Warm starting

Motivated by the observation that GANs tend to be unstable in early rounds of training, previous work [ZJW18] has proposed to warm-start the DP-GAN by treating a small portion of the data ($\approx 2\%$) as public and using them to train the GAN non-privately. Then the final parameters of the non-private GAN can be used as starting parameters in the first iteration of DP-GAN training. This will ensure that the DP-GAN starts from a position that is already approximately correct, and thus will require fewer rounds of training. [ZJW18] showed that this techniques improves accuracy by about 15% for standard statistical measures in machine learning.

However, in many real-world applications, releasing even a small subset of sensitive data may raise legal or ethical concerns. Instead, we propose using relevant publicly available datasets for warm starting. For example, we can use publicly available 1940 Census data to warm start the training on 2010 Census data, or a published clinical trial dataset for training on a medical-related dataset. 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.

3.3 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 (see Section 3.1). Therefore, directly applying the 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.

To achieve this, we keep track of our accumulated privacy loss using the privacy accountant of [ACG$^+$16], which is designed for computations where the randomization and privacy loss of mechanism's current iteration depends on the outcome of past iterations. Intuitively, this allows the analysis of composing probability ratios directly on the sequence of possible outcomes over iterations, rather than composing the sequence of privacy losses over iterations. For a formal definition of the privacy accountant and the relationship between its composition guarantees and regular (ϵ, δ)-differential privacy, we refer the interested reader to [ACG$^+$16].

In practice, the accumulated privacy loss over training is the key to deciding when a pre-specified privacy budget has been reached and training should halt.

3.4 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. All modifications presented in this section are on the underlying neural network structure, so they can be implemented straightforwardly in our DP-GANs framework.

Continuous data. Real-valued data is easily handled by neural networks and thus DP-GANs as well. This will make regression tasks among the easiest for our DP-GAN to handle without much further optimization. See, for example, the success of using DP-GANs to generate clinical data, whose features are blood pressures of subjects over multiple visits [BJWWG17].

Binary data. It has been generally observed that neural networks can handle binary data (both as input and output, as in our case of synthesizing data task). Examples of techniques to incorporate binary data include adding a perceptron activation function in the output layer to generate binary data or Gumbel-softmax function as an approximation to multinomial distribution. It is folklore that neural networks are particularly well-suited for 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 [CBM+17] 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 [JGP16, KHL16] 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 [WWW+17]. 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.

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 [BJWWG17]. 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. For example, in clustering, α_t can be defined as the negative of the additional objective value from using G^t instead of holdout set. We can then choose Q by Report Noisy Argmax as described above.

Alternatively, we can use statistical scores (e.g., Jensen-Shannon scores, as described in the answers to questions) which does not require any task to be specified. This approach works for labeled and unlabeled data, and allows for accuracy improvements for unknown research questions as well. Then, we can proceed to choose Q as before, using Report Noisy Argmax on the statistical scores instead of α_t.

5 Our DP-GAN algorithm

In this section, we combine all the improvements proposed in Sections 3 and 4 into our proposed solution, given in Algorithm 3.

Algorithm 3 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 η_y, η_w; minibatch size m; minibatch size for public data m_{public}; noise scale σ; number of parameter groups l; privacy budget (ϵ_0, δ_0).

while y has not converged **do**

for k steps **do**

sample minibatch of m_{public} noise samples $\{z_i\}_{i=1}^{m_{\text{public}}}$ from p_z

sample minibatch of m_{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

$$\{(G_j, c_j)\}_{j=1}^{l} \leftarrow \text{Weight-Clustering}(l, \{\overline{g}_w^{(i)}\}_{i=1}^{m_{\text{public}}}) \tag{1}$$

sample minibatch of m data samples $\{x_1, \ldots, x_m\}$ from p_{data}

sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ from p_z

compute stochastic gradient g_w

clip gradient g_w according to grouping (G_j, c_j) obtained from weight clustering

$$\begin{aligned} g_{w,(j)} &\leftarrow (g_w \cap G_j), && \text{for } j = 1, \ldots, l \\ g_{w,(j)} &\leftarrow g_{w,(j)} \min(1,\ c_j / \|g_{w,(j)}\|), && \text{for } j = 1, \ldots, l \end{aligned}$$

add corresponding noise in the clipped gradient groups

$$g_{w,(j)} \leftarrow g_{w,(j)} + \mathcal{N}(0, \sigma c_j \mathbf{I}),\ \text{for } j = 1, \ldots, l$$

update discriminator, D_w, by ascending along stochastic gradient

$$\begin{aligned} w_j &\leftarrow w_j + \eta_w g_{w,(j)},\ \text{for } j = 1, \ldots, l \\ w &\leftarrow \{w_j\}_{j=1}^{l} \end{aligned}$$

end for

sample minibatch of m noise samples $\{z_1, \ldots, z_m\}$ 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

$$\delta \leftarrow \exp\Big\{ - \frac{\sigma^2 \epsilon_0^2}{c_2^2 q^2 T k} \Big\}$$

if $\delta > \delta_0$ **then**

break

end if

end while

The weight clustering subroutine is used in line (1) of Algorithm 3. This subroutine is given below in Algorithm 4.

Algorithm 4 Weight-Clustering

Input parameters: number of groups l; stochastic gradients $\{\overline{g}_w^{(i)}\}_{i=1}^{m_{\text{public}}}$
compute average gradient

$$g = \frac{1}{m_{\text{public}}} \sum_{i=1}^{m_{\text{public}}} \overline{g}_w^{(i)} \tag{2}$$

initialize groups as $\mathcal{G} \leftarrow \{g_j, c_j\}$ where g_j are elements of vector g and $c_j \leftarrow |g_j|$ is the average gradient
while $|\mathcal{G}| > k$ **do**
compute G, G' groups which have closest possible c's

$$G, G' \leftarrow \arg \min_{G,G' \in \mathcal{G}} \max \left(\frac{c(G)}{c(G')}, \frac{c(G')}{c(G)} \right)$$

merge G, G' to get updated set $\mathcal{G}$

$$\begin{aligned} G'' &\leftarrow G \cup G' \\ \mathcal{G} &\leftarrow \mathcal{G} \setminus \{G, G'\} \cup G'' \\ c(G'') &\leftarrow \sqrt{c(G)^2 + c(G')^2} \end{aligned}$$

end while
return $\mathcal{G}$

References

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[BJWWG17] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, and Casey S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. bioRxiv preprint 159756, 2017.

[CBM+17] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Proceedings of Machine Learning for Healthcare Conference*, pages 286–305, 2017.

[CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, 2008.

[GPAM+14] 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 27*, pages 2672–2680, 2014.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. arXiv preprint 1611.01144, 2016.

[KHL16] Matt J Kusner and José Miguel Hernández-Lobato. GANs for sequences of discrete elements with the Gumbel-softmax distribution. arXiv preprint 1611.04051, 2016.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1097–1105, 2012.

[MDH12] Abdel-rahman Mohamed, George E. Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *Transactions on Audio, Speech, and Language Processing*, pages 14–22, 2012.

[WWW+17] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. arXiv preprint 1711.08267, 2017.

[XLW+18] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. arXiv preprint 1802.06739, 2018.

[ZBH+16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. arXiv preprint 1611.03530, 2016.

[ZJW18] Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model. arXiv preprint 1801.01594, 2018.