CSC2412 Project: DPGAN and Model Inversion

Jason Tang

April 15, 2023

Abstract

The growing concern over privacy in data-driven applications has led to the development of various privacy-preserving techniques, including Differential Privacy for Model Inversion (DPMI) [3]. DPMI has shown promise in generating synthetic data for privacy-preserving applications using Generative Adversarial Networks (GANs), which often suffer from issues such as mode collapse and long training times. In this project, we explored the use of autoencoders and Variational Autoencoders (VAEs) as potential alternatives to GANs in the DPMI framework. We conducted experiments to compare the performance of autoencoders and VAEs with GANs in terms of stability, privacy, and utility, using evaluation metrics such as Inception Score (IS) and Fréchet Inception Distance (FID). Overall, our findings suggest that autoencoders and VAEs offer simplicity and stability compared to GANs, and show potential as viable GAN alternatives in situations when GANs are too unstable or expensive to train.

1 Introduction

With advances in information technology and mobile computing, increasing amounts of consumer data are being collected and utilized to train machine learning models. Though the widespread utility of these models is undeniable, both ethical and legal concerns arise due to the combination of sensitive portions of the data, and the susceptibility of large machine learning models to membership attacks [19]. Differential privacy (DP) presents a solution to these data sensitivity issues in the form of formal privacy guarantees, and is commonly applied by adding gaussian noise during the training process, such as in DP-SGD [1].

These methods have become increasingly popular for privately training and releasing models for downstream tasks, but lack the ability to release any datasets, as the original private data remains unaltered. One alternative is to release a synthetic dataset or generative model under differential privacy to train non-private downstream models, as the privacy of the original sensitive data is protected through DP post-processing. This dataset releasing method is simple to integrate into standard non-private training pipelines as a data input modification, can be combined with other datasets to train stronger models, and allows the safe transfer of sanitized data to other researchers for the development of new technologies.

Within deep learning, generative models such as autoencoders and generative adversarial networks (GANs) are often released to generate synthetic data for downstream applications. In this project, we will explore several proposed differentially private generative models and to evaluate the empirical effects of potential improvements.

2 Related Work

2.1 DP-SGD

Differential privacy (DP) presents a solution to these data sensitivity issues in the form of formal privacy guarantees, and is commonly applied by adding gaussian noise during the training process, such as in DP-SGD [1]. In this seminal paper, Abadi et al. present a differentially private method for training deep neural

networks by adding random noise to clipped gradient updates, both of which are necessary for provable privacy guarantees. Since the noise is gaussian, this leads to (ϵ, δ)-DP, which is defined as follows:

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

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

Additionally, we define sampling probability $q = L/N$, where L is the batch size of examples averaged before adding noise, and N is the total number of examples.

They also introduce the moment account method, which provides a tighter worst case privacy bound compared to strong composition, allowing for lower privacy costs during each iteration. Specifically, they reduce the strong composition privacy bound $(\mathcal{O}(q\epsilon\sqrt{T\log(1/\delta)}), Tq\delta)$ to the moments accountant bound $(\mathcal{O}(q\epsilon\sqrt{T}), \delta)$. Which is significant as δ is usually very small, and $T \gg 1/q$ as each training example is used multiple times. The resulting (ϵ, δ) differential privacy is useful as it provides graceful degradation, robustness to auxiliary information, and can easily be combined with other methods due to the composition theorem.

2.2 Tempered Sigmoids

One issue with using DP-SGD to train deep learning models is the commonly used ReLU activation function and its unbounded range. Papernot et al. [15] observe that the lack of a bound on model activations leads to exploding gradients, which, when combined with the gradient clipping of DP-SGD, leads to low utility gradient updates. As such, the authors propose using a family of bounded functions called tempered sigmoids instead of ReLUs as activation functions to enhance performance. They demonstrate that the use of tempered sigmoids allows for explicit control of the gradient norm and model sensitivity. This is evidenced by empirical results which demonstrate faster convergence and the reduction of negative impacts caused by clipping and adding noise to large gradients.

2.3 Feature Extraction

Tramèr and Boneh [20] demonstrate an effective baseline method based on extracting non-learned handcrafted features from private images using Scatternet [13] (a SIFT-like feature extractor). Due to the reduction in dimensionality, this method requires far fewer iterations to converge, and as such, allows practitioners to inject less noise into each gradient step for the same differential privacy guarantees. As a result of this baseline, the authors suggest that differentially private learning can achieve model performance comparable to non-private learning by exploring two potential directions: designing non-learned features or pretraining on public data.

As such, most of the recent methods developed for private image classification focus on utilizing pretrained weights learned on public data and fine tuning only the last layers in a private manner [11] [10]. One such method is presented by Mehta et al. [10], which explores more sophisticated second-order optimization methods such as Newton's method to privately tune the last layer of a pretrained model learned on public data. In particular, they introduce a training method known as DP-FC, which utilizes feature covariance instead of the hessian as second order information, and performs especially well with smaller epsilon values (< 1). Another recent method by Mehta et al. is "Large Scale Transfer Learning for Differentially" [11], which explores using the LAMB optimizer for last layer fine tuning, which is intended to perform well with large batch sizes, and analyzes single step training in the full batch setting.

2.4 DPGAN

The popularity of GANs has surged in recent times due to their remarkable capacity to hallucinate lifelike images. However, the highly expressive nature of non-private GANs makes them susceptible to memorizing the training data. To address this, Xie et al. propose a differentially private generative adversarial network

(DPGAN) [21] that protects the privacy of training data using a WGAN architecture [2], which solves the following minimax game:

$$\min_{G} \max_{\mathbf{w} \in W} \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[f_{\mathbf{w}}(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[f_{\mathbf{w}}(G(\mathbf{z}))] \tag{1}$$

where G is the generator, $f_{\mathbf{w}}$ is the discriminator ($\mathbf{w}$ are its parameters), $p_{data}(\mathbf{x})$ is the real data distribution, $p_{\mathbf{z}}(\mathbf{z})$ is a gaussian distribution, and W is the set of all possible parameters for $\mathbf{w}$. This results in the following losses which are minimized:

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[f_{\mathbf{w}}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[f_{\mathbf{w}}(G(\mathbf{z}))] \tag{2}$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[f_{\mathbf{w}}(G(\mathbf{z}))] \tag{3}$$

Additionally, WGAN sets the constraint that all $f_{\mathbf{w}}(x)$ are K-Lipschitz (with respect to x) for some K, i.e. their weights are clipped.

As a result of this weight clipping, the authors are able to replace the DP-SGD gradient clipping with explicitly defined gradient bounds from weight clipping (Lemma 3.5 in the paper):

$$\left\| g_{\mathbf{w}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) \right\| \leq 2 c_p B_\sigma B_{\sigma'}^2 \sum_{k=1}^{H-1} m_k m_{k+1} \tag{4}$$

where $g_{\mathbf{w}}$ is the gradient of the discriminator's parameters $\mathbf{w}$ with respect to the Wasserstein loss, c_p is the weight clipping constant, B_σ is the range bound on activation functions ($\sigma(\cdot) \leq B_\sigma$), $B_{\sigma'}$ is the bound on activation derivatives ($\sigma'(\cdot) \leq B_{\sigma'}$), m_k is the number of neurons in layer k, and H is the number of non-input layers in the discriminator (assuming a fully connected network).

Their proposed algorithm guarantees (ϵ, δ) differential privacy using the Moments Accountant from DP-SGD [1] and is shown to be effective in generating reasonable MNIST images, albeit at relatively high ϵ's (in the range [9.6, 29.0]). Additionally, the use of WGAN over a traditional GAN [5] using Jensen-Shannon divergence helps stabilize training and generally avoids issues such as mode collapse.

2.5 DP Model Inversion

Despite the stability improvements, learning a GAN from scratch is still a lengthy process which requires hundreds of thousands of iterations to converge, which requires large amounts of noise to gradients to maintain the same privacy guarantees. One way to improve the training efficiency of complex DP-GANs is through dimensionality reduction [12]. Chen et al. explore this idea in "Differentially Private Generative Adversarial Networks with Model Inversion" (DPMI) by first publicly training an Improved-WGAN [6], which uses gradient norm penalty instead of weight clipping. This is done by changing the losses to:

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[f_{\mathbf{w}}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[f_{\mathbf{w}}(G(\mathbf{z}))] + \lambda \left(\|\nabla_{\hat{\mathbf{x}}} f_{\mathbf{w}}(\hat{\mathbf{x}})\|_2 - 1 \right)^2 \tag{5}$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[f_{\mathbf{w}}(G(\mathbf{z}))] \tag{6}$$

where $\hat{\mathbf{x}} = c\mathbf{x} + (1-c)G(\mathbf{z})$, $c \sim U[0,1]$ is random number, and λ is the gradient penalty coefficient. This imposes a soft constraint towards 1-Lipschitz on the discriminator.

The authors then perform model inversion using gradient ascent on the resulting public generator to compute the closest latent space representation of the private data points [3]. To do so, they follow this optimization process (Equation (4) in the paper):

$$\begin{aligned} z_s &= \text{argmin}_z \left\| G(z) - \mathbf{x} \right\|^2 \\ &\text{s.t. } P_Z(z) \geq P_Z(z_0) \end{aligned} \tag{7}$$

where $z_0 \sim P_Z = \mathcal{N}^d(0, I)$, where d is the size of the latent dimension. This convex constraint ensures that the latent vector z_s is of a quality comparable to $G(z_0)$. The solution to this optimization problem is found through projected gradient ascent.

These private latent vectors are then used to learn a standard DP-GAN in the lower-dimensional latent space, leading to fewer iterations necessary for convergence and less additive noise for the same privacy guarantees. The final DP-GAN generator can then be released to generate synthetic latent vectors, which can then be inputted into the original public GAN to generate synthetic images approximating to those within the private dataset.

The authors also compare the standard DP-SGD paradigm to Private Aggregation of Teacher Ensembles (PATE) [14], which uses multiple models trained on disjoint subsets of sensitive data as teachers for a student model. The student model learns to predict an output chosen by noisy voting among all of the teachers, and cannot directly access an individual teacher or the underlying data or parameters. The student model then learns to predict outputs chosen through noisy voting among multiple teachers, without having direct access to teacher models or underlying data. As a result, this method is able to impose only weak assumptions on the teacher model architectures or training processes, which provides much more flexibility compared to DP-SGD based methods. However, PATE-based training methods tend to be data inefficient because they require training multiple teacher models on separate sets of data, which can be challenging with small datasets commonly encountered in real-world scenarios. As a result, the authors of DPMI chose to adopt DP-SGD despite the limitations imposed on the potential models.

2.6 DP Autoencoders

In contrast to other studies that primarily focus on GAN variants, Chen et al. [4] investigate the robustness of differentially private autoencoders and variational autoencoders [9] against model inversion and GAN-based adversarial attacks. They propose two models: an autoencoder-based generative model (AuGM), and a variational autoencoder generative model (VaeGM). To generate labeled synthetic data in VaeGM, the authors adopt a strategy of privately training separate variational autoencoders (VAEs) for each class using DP-SGD [1]. The use of VAEs allows for infinite data generation by simply sampling random Gaussian noise, but empirical results show that it is unstable when trying to generate high-quality samples. AuGM works by first privately training an autoencoder using DP-SGD and then using the encoder of the autoencoder to synthesize information from the private data into the generated outputs. This provides stronger protection against model inversion and GAN-based attacks compared to VaeGM, but requires downstream users to have access to a large amount of related public data to pass through the encoder to generate the synthetic data.

3 Methodology

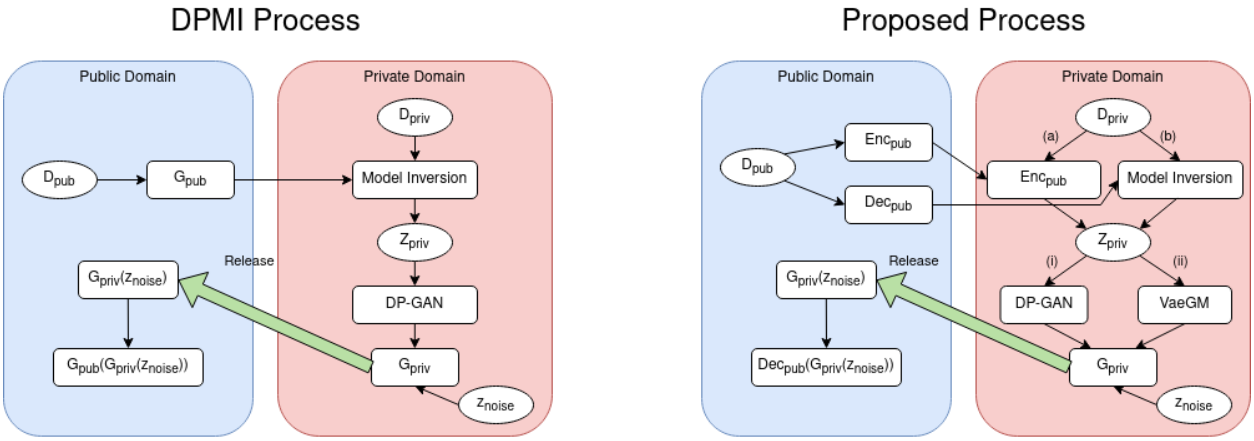

Figure 1: Diagram of the DPMI framework and our proposed modifications.

In this project, we will explore several proposed variations to the DPMI framework and empirically assess their performance (See Fig. 1 for the setup). The main difference is the use of a publicly trained autoencoder (AE) instead of a GAN, which is intended to alleviate some of the issues found in GAN training,

Public Method	Latent Extraction Method	Private Learning Method	Abbreviation
GAN	Gradient Ascent	DP-GAN	wgan-GAN (DPMI)
GAN	Gradient Ascent	VaeGM	wgan-VAE
Autoencoder	Encoder	DP-GAN	ae-enc-GAN
Autoencoder	Encoder	VaeGM	ae-enc-VAE
Autoencoder	Gradient Ascent	DP-GAN	ae-grad-GAN
Autoencoder	Gradient Ascent	VaeGM	ae-grad-VAE

Table 1: Naming conventions for the different methods.

as well as provide the ability to simply encode the private data with the public encoder to get the latent representations, or use the DPMI method of gradient ascent using the public decoder (Fig. 1 (a) and (b), respectively). Additionally, besides the DP-GAN [21] used to generate synthetic latent vectors, we also consider a single-class version of VaeGM [4] since we don't require labels (Fig. 1 (i) and (ii), respectively).

Overall, the aim of this project is to explore substituting GANs with simpler, more stable models such as autoencoders and VAEs which are less expressive, but avoid training issues such as mode collapse and long learning times required to generate higher quality results. Autoencoders and VAEs were chosen as they have been shown to have success in DP applications [4] before, and VAEs are similar to GANs in their ability to generate an infinite amount of synthetic data once trained. Through this, we hope to provide an easier to train and more stable alternative to the DPMI framework, which can be used to generate synthetic data for applications where training GANs would be unstable.

4 Experiments

In our implementation, we mainly utilize PyTorch [1] for model training and data management, and Opacus [2] [22] for privacy accounting. Our implementation is available on GitHub: `https://github.com/JasonTang99/csc2412_project`, and our experiment results are available on Google Drive.

4.1 Dataset

We run all experiments on the MNIST dataset, which consists of 60,000 28x28 pixel grayscale images of handwritten digits. For data partitioning, we follow the method used in DPMI [3]:

- Labeling dataset: 1/3 of training data (20,000 images)
- Public dataset: 1/2 of classes (odd digits) in the remaining data (20,388 images)
- Private dataset: 1/2 of classes (even digits) in the remaining data (19,612 images)
- Test dataset: original test data (10,000 images)

We maintain the same random partition throughout all experiments.

4.2 Metrics

We utilize the common metrics of Inception Score (IS) [17] and Fréchet Inception Distance (FID) [7] to empirically evaluate the quality of the generated data. Inception Score (IS)[17] is a metric that measures the quality and diversity of synthetic images which utilizes features extracted from a pre-trained Inception V3 network (larger is better). Frechet Inception Distance (FID) [7] is an improvement upon IS which that measures the distance between the distributions of real and synthetic images (smaller is better).

We refer to an implementation of Inception Score [3] and Fréchet Inception Distance [4] for our experiments.

[1] https://pytorch.org/
[2] https://opacus.ai/
[3] https://github.com/sbarratt/inception-score-pytorch
[4] https://github.com/mseitzer/pytorch-fid

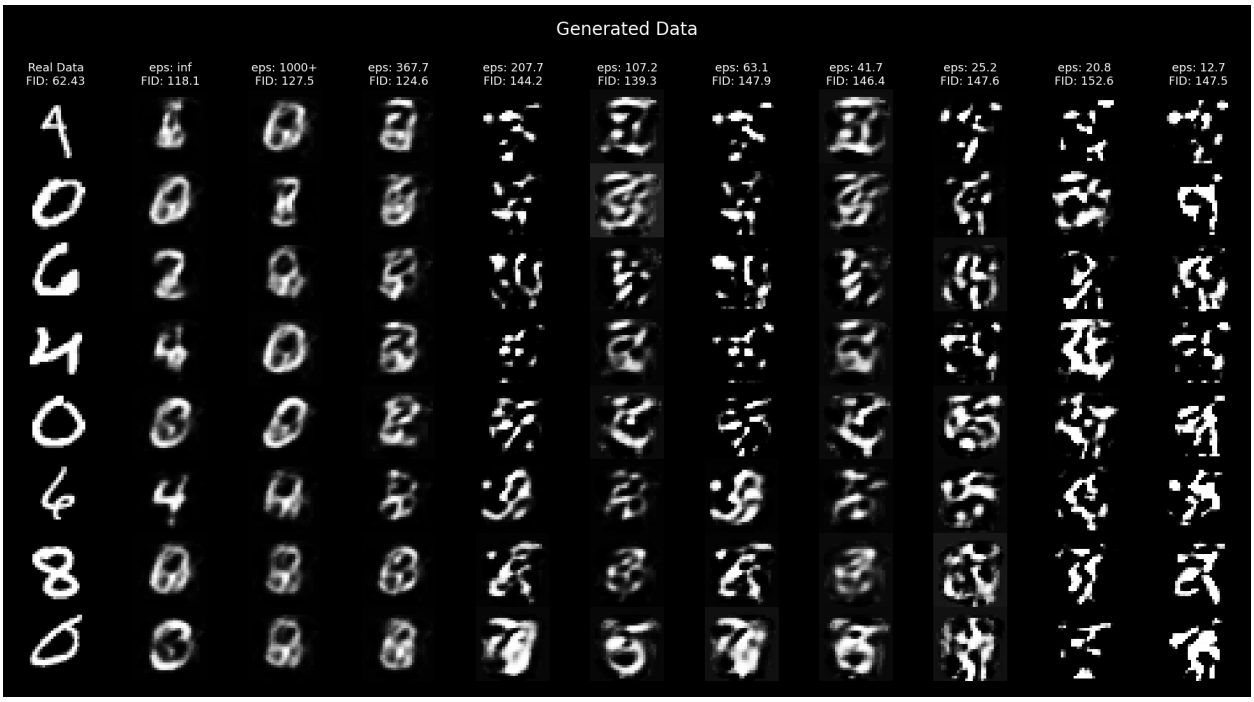

Figure 2: Generated data at different values of ϵ using the ae-grad-VAE model.

4.3 Hyperparameters

We explore a range of hyperparameter settings detailed for all steps in the appendix (Table 2). Due to limited time and computational resources, we were unable to explore a wider set of hyperparameters for performances comparable to the original DPMI framework. However, we believe that our results are still useful in demonstrating the stability improvements of our proposed modifications to the DPMI framework.

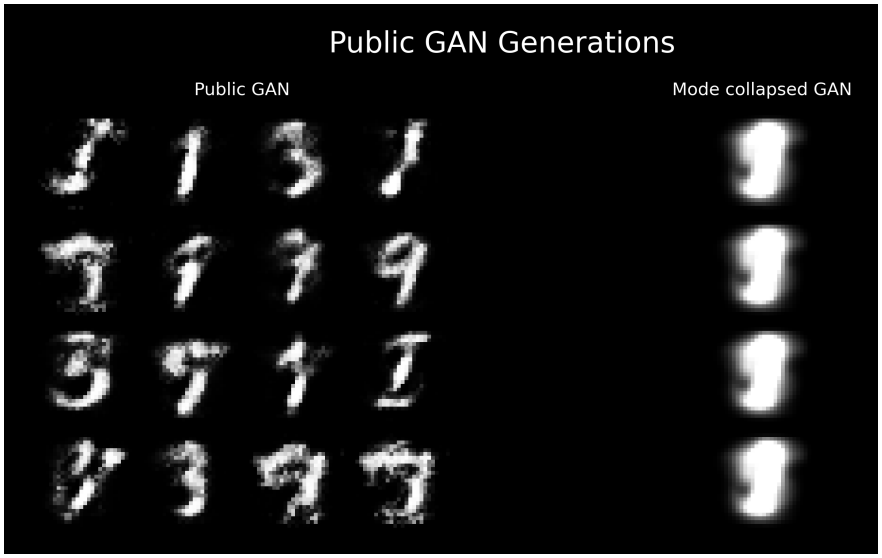

Figure 3: Samples generated by the public GAN.

4.4 GAN vs. AE Training

Throughout our hyperparameter searches and experiments, we found it difficult to get publicly trained GANs to converge, but those that did were able to generate reasonable results for the simple fully-connected generator models we used (Fig. 3). We found that most of our attempt to publicly train GANs failed, and all of our experiments with DP-GAN (DPMI, ae-enc-GAN, ae-grad-GAN) failed to converge (Fig. 10). This could be potentially due to bugs in our code but we believe it is likely a result of insufficient hyperparameter search combined with the instability of GAN training. Moreover, each GAN training experiment would take between 2-3 hours to complete, which made it difficult to run a large number of experiments.

Comparatively, we were able to train each version of the public autoencoder in a single experiment in under 1 hour, and each experiment with the private VaeGM took less than 10 minutes. This is likely due to the fact that autoencoders and VAEs are much simpler models than GANs, and are far more stable to train at higher learning rates.

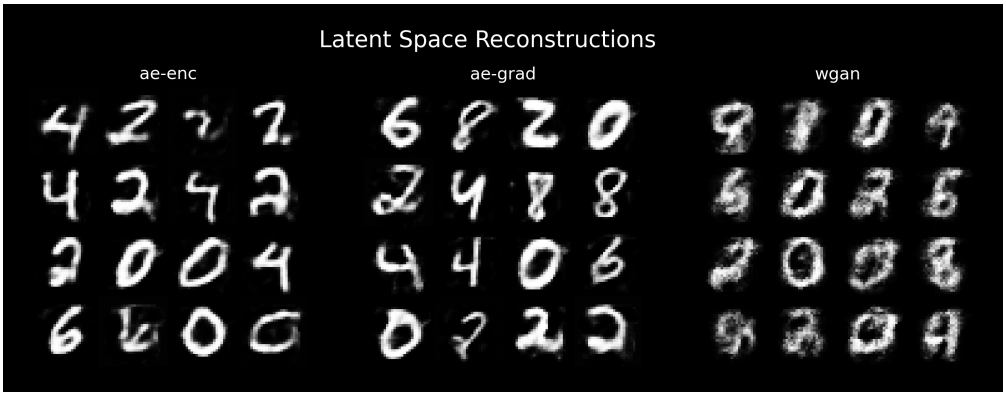

Figure 4: Sample reconstructions from different latent extraction methods.

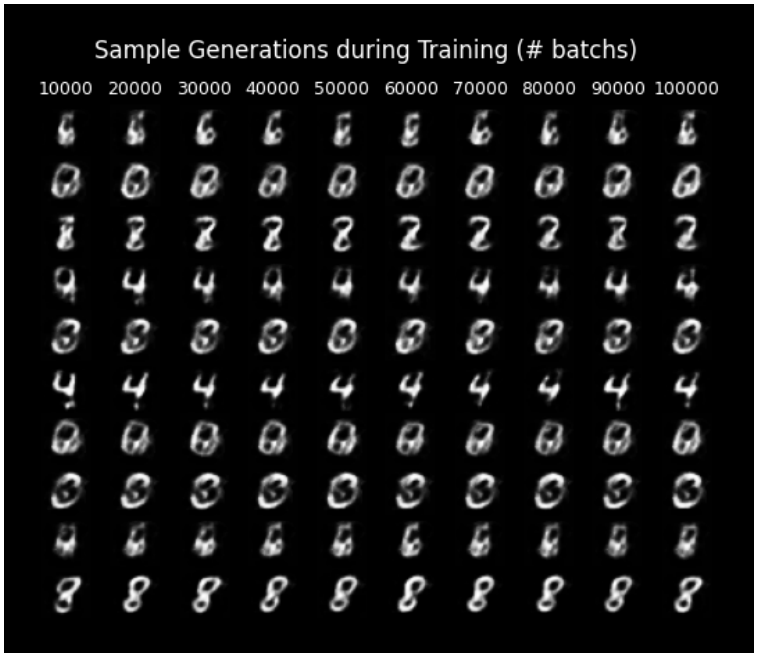

Figure 5: Generated images during the training of the ae-grad-VAE model. $\epsilon = \infty$.

4.5 Latent Extraction

We found that the gradient ascent (ae-grad) algorithm was able to find latent representations which generate high quality reconstructions (Fig. 4). We can also see that the projected gradient ascent (wgan-grad) method found latents with reasonable quality, given the poorly hyperparameter tuned public GAN model. This resulted in our later wgan-grad-VAE models failing to produce anything other than a single mode, even without privacy. We believe that with improved public GAN training, the wgan-grad-VAE method would be able to produce results comparable to those of ae-grad-VAE.

We use 200,000 iterations for both algorithms in all experiments (Fig. 8). Both of these latent extraction methods was only ran once for the private data and took under 30 minutes to complete.

Additionally, we found that the reconstructions from private data encoded using the public autoencoder (ae-enc) were of fairly high quality (Fig. 4), and took less than a minute to complete. However, we also found it difficult to train a VaeGM model (ae-enc-VAE) on these encoded latents, and the models were only able to output a single mode. We believe that this may be due to the encoded latents being in an non-well-behaved space (e.g. unsmooth), whereas gradient ascent methods are more likely to find a well-behaved local minimum.

Since none of our *-GAN methods were successful, this leaves us with only a functional ae-grad-VAE model, which we will now analyze (Fig. 5).

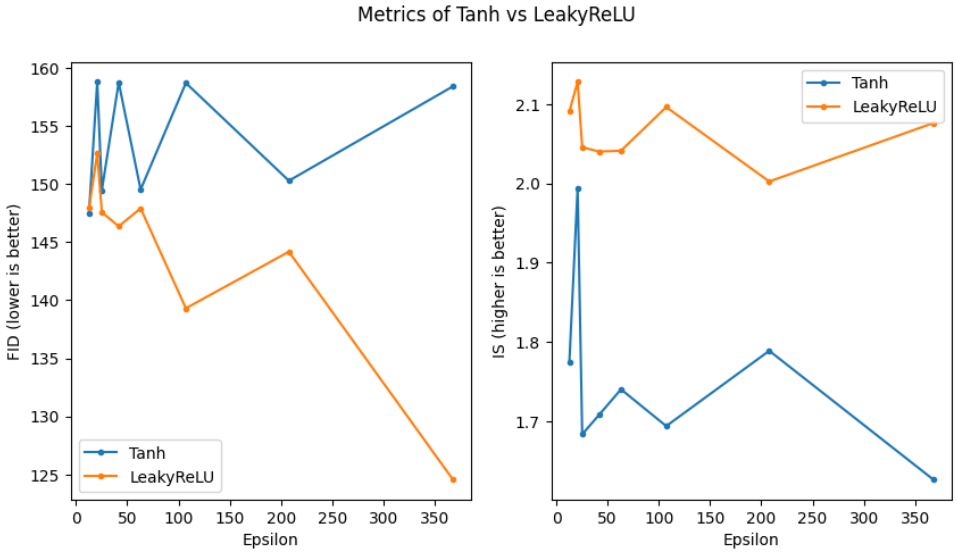

Figure 6: Comparison between tanh and LeakyReLU activation functions.

4.6 Activation Functions

In our ae-grad-VAE experiments, we compared the performance of tanh and LeakyReLU (Slope of 0.2) activation functions. Fig. 6 shows the performance of the best tanh and LeakyReLU models at each ϵ. We found that the LeakyReLU activation function was able to produce better results at higher ϵ settings, but tanh performed comparably to LeakyRelu at lower ones, with a tanh model slightly beating the best LeakyReLU model at the lowest ϵ in terms of FID.

We believe that this is due to the suggested idea from Papernot et al. [15] that, since the tanh function has bounded range, it is better suited for small gradient norm clipping values, which is usually the case of smaller ϵ settings. Based on this, we think that if we explored lower ϵ settings, we would find that tanh would likely perform comparably or even better than LeakyReLU.

A minor drawback of tanh is the computational cost of computing the gradient of the tanh function, which is more expensive than the gradient of the LeakyReLU function which is only ever 1 or 0.2. We found that tanh generally took about 10-20% longer to train than LeakyReLU, but this was not a significant factor in our experiments.

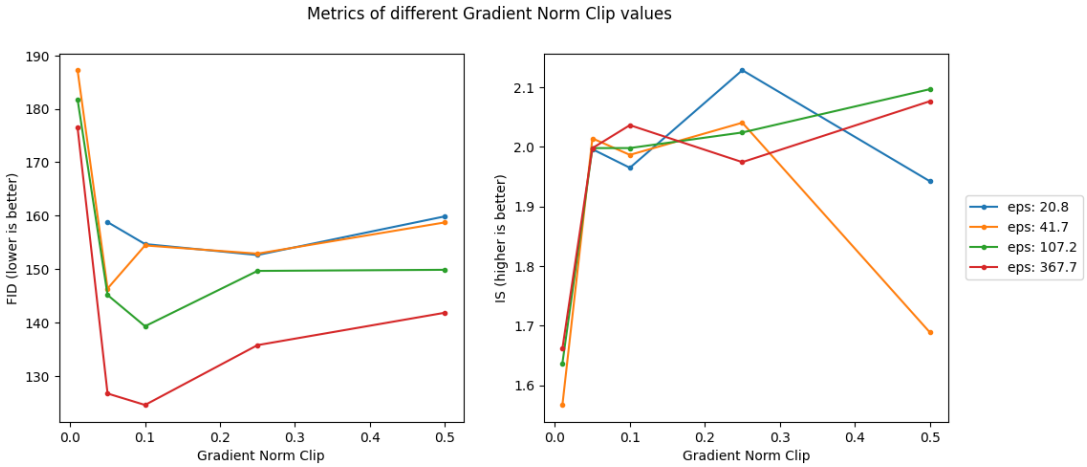

Figure 7: Comparison between different Gradient Norm Clipping values.

4.7 Gradient Norm Clips at Different ϵ Settings

We also analyze the difference in which gradient norm clipping value is best at each epsilon level (See Fig. 7). We found that, for our hyperparameter settings, the best gradient norm clipping value was usually between 0.05 and 0.1. Note that performance drops sharply at a value of 0.01, and degrades more slowly as the clip value increases.

This hyperparameter was very important to tune, as the gradient clip value determines the sensitivity of the entire model, which in turn, affects how much noise we need to add to the gradients. In our tuning, we aimed to decrease this value as much as possible in order to add less noise without sacrificing too much utility in the resulting gradients.

5 Discussion and Conclusions

In this project, we explored several variations of the Differential Privacy for Model Inversion (DPMI) framework [3], utilizing autoencoders and VAEs in place of GANs for generating synthetic data. Our results demonstrated the instability of training GAN-based methods, and explored a more stable approach to generate synthetic data while maintaining privacy, as measured by Inception Score (IS) and Fréchet Inception Distance (FID).

One of the main advantages of using autoencoders and VAEs in the DPMI framework is their simplicity and stability compared to GANs, which are often difficult to train as a result of mode collapse issues and long training times. Another advantage of the VAE in particular is the ability to match the GAN in its ability to generate an infinite amount of synthetic data once trained, which is critical in downstream applications requiring large amounts of data such as data analysis and machine learning.

As is the case with other DP methods, it is important to consider the trade-offs between privacy and utility. As the level of privacy and added noise increases, the utility of the generated synthetic data will decrease, as we can see in 2. Therefore, finding the right balance between privacy and utility is an important decision to make in applying our methods in real-world scenarios.

Our results showed that autoencoders and VAEs can potentially be viable alternatives to GANs in the DPMI framework, providing simplicity and stability in generating synthetic data for privacy-preserving applications, especially in applications where there is insufficient data or resources to train the GANs required in vanilla DPMI.

6 Open Problems and Future Work

The main path of improvement for the proposed DPMI modifications would be through the exploration of more complex models throughout all the experiments, as we largely used simple 1-2 layer fully connected neural networks. Additionally, a deeper and wider hyperparameter search across the different settings will likely result in far better metrics comparable to those in the original DPMI paper. Moreover, it would be beneficial to see an expanded set of metrics such as downstream model accuracy or robustness to model inference attacks [19] [16], which better captures the utility of the method in downstream applications compared to IS and FID.

In the context of the DPMI framework and DP generative models in general, scalability and efficiency of privacy-preserving methods in real-world deployments are significant open problems. One popular method of achieving this is through Collaborative Deep Learning [18], a process which allows multiple parties to jointly learn a model without sharing their individual datasets. However, there have been proposed attacks which prove to be effective against these collaborative deep learning methods [8]. Further research is needed to develop distributed, federated, or decentralized methods capable of training robust generative models in a scalable manner, especially on large datasets. Classical cryptographic methods like multiparty computation or fully homomorphic encryption could potentially be used as solutions, but they are computationally costly and still have vulnerabilities [8]. Therefore, exploring different granularities of differential privacy, such as at the user or device level, could be a potential solution for privacy-preserving collaborative learning.

Another open problem is the interpretability and explainability of DP generative models. While these models can generate synthetic data with privacy guarantees, comprehending the relationship between the generated data and the original data, as well as the underlying data distribution, can be challenging. Developing techniques for interpreting and explaining the outputs of these models while still preserving the privacy of private data could greatly enhance the trust and usability of these methods in sensitive real-world applications where concerns about biases are prominent. Addressing this challenge could contribute to the responsible and ethical use of differential privacy generative models in practical settings.

Lastly, as the adoption of differential privacy grows in popularity as a means of providing provable privacy guarantees, there is an increasing need for practical guidelines and best practices for real-world deployments of DP generative models. While these models are valuable for privacy-preserving data generation, their practical implementation and hyperparameter settings can be complex and challenging to comprehend. Establishing guidelines for model selection, hyperparameter tuning, and privacy budget allocation can assist practitioners in effectively deploying these models with a clear understanding of the privacy guarantees they offer, avoiding a false sense of security.

References

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

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *ArXiv*, abs/1701.07875, 2017.

[3] D. Chen, S.-C. S. Cheung, C.-N. Chuah, and S. Ozonoff. Differentially private generative adversarial networks with model inversion. *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2021.

[4] Q. Chen, C. Xiang, M. Xue, B. Li, N. Borisov, D. Kaafar, and H. Zhu. Differentially private data generative models. *ArXiv*, abs/1812.02274, 2018.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[8] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning, 2017.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022.

[10] H. Mehta, W. Krichene, A. Thakurta, A. Kurakin, and A. Cutkosky. Differentially private image classification from features. *ArXiv*, abs/2211.13403, 2022.

[11] H. Mehta, A. Thakurta, A. Kurakin, and A. Cutkosky. Large scale transfer learning for differentially private image classification. *ArXiv*, abs/2205.02973, 2022.

[12] B. Neyshabur, S. Bhojanapalli, and A. Chakrabarti. Stabilizing gan training with multiple random projections. *ArXiv*, abs/1705.07831, 2017.

[13] E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2865–2873, 2014.

[14] N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *ArXiv*, abs/1610.05755, 2016.

[15] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. In *AAAI Conference on Artificial Intelligence*, 2020.

[16] C. Park, Y. Kim, J.-G. Park, D. Hong, and C. Seo. Evaluating differentially private generative adversarial networks over membership inference attack. *IEEE Access*, 9:167412–167425, 2021.

[17] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016.

[18] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[19] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2016.

[20] F. Tramèr and D. Boneh. Differentially private learning needs better features (or much more data). *ArXiv*, abs/2011.11660, 2020.

[21] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. Differentially private generative adversarial network. *ArXiv*, abs/1802.06739, 2018.

[22] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

7 Appendix

Hyperparameter	**Values**
Fixed for all experiments	
Batch Size	64, 32
Activation	Tanh, LeakyReLU(0.2)
Public Latent Size	100
Optimizer	Adam(0.5, 0.9, weight decay 1e-6)
Public GAN	
Model Architecture	[[256], [128], [128, 128]] FC hidden layers, 3 Layer CNN (32, 4, 2, 1)
Learning Rate	5e-05, 5e-04
Discriminator Update Ratio	5:1, 3:1
Generator Updates	5e5
GAN Variants	WGAN, WGAN-GP
Public Autoencoder	
Model Architecture	[128] FC hidden layers
Learning Rate	5e-04
Epochs	1000
Private DP-GAN (WGAN)	
Model Architecture	[[128], [96], [32], [16], [16, 12], [12, 4, 4]] FC layers
Learning Rate	5e-06, 1e-05, 5e-05
Noise Multiplier	0.0, 0.05, 0.1, 0.2
Clipping Norm	0.005, 0.01
Discriminator Update Ratio	5:1, 3:1
Private Latent Size	32, 64
Private VaeGM	
Model Architecture	[64] FC hidden layers
Learning Rate	0.005, 0.01, 0.02
Noise Multiplier	0.0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
Clipping Norm	0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5
Batch Updates	1000, 10000, 48000, 100000
Private Latent Size	32

Table 2: Hyperparameters used in the experiments.

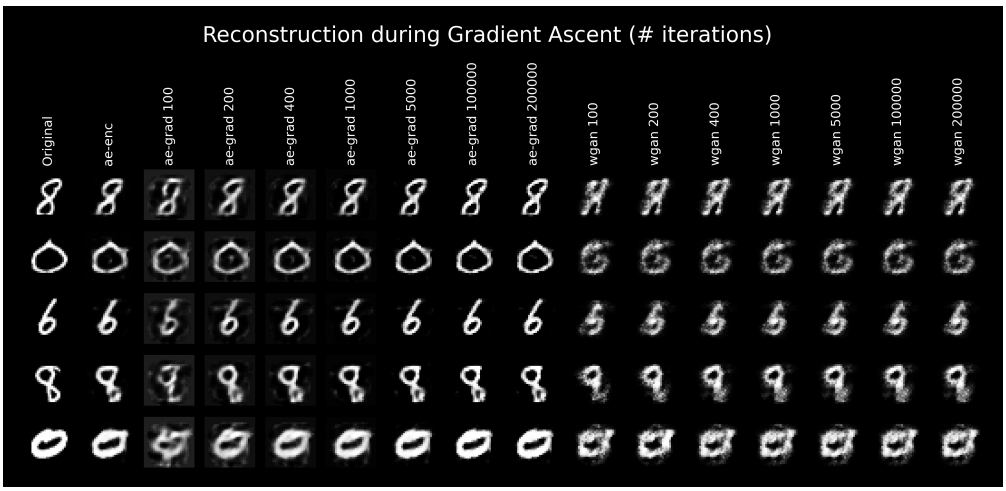

Figure 8: Reconstructions at different iteration steps of the gradient ascent algorithm.

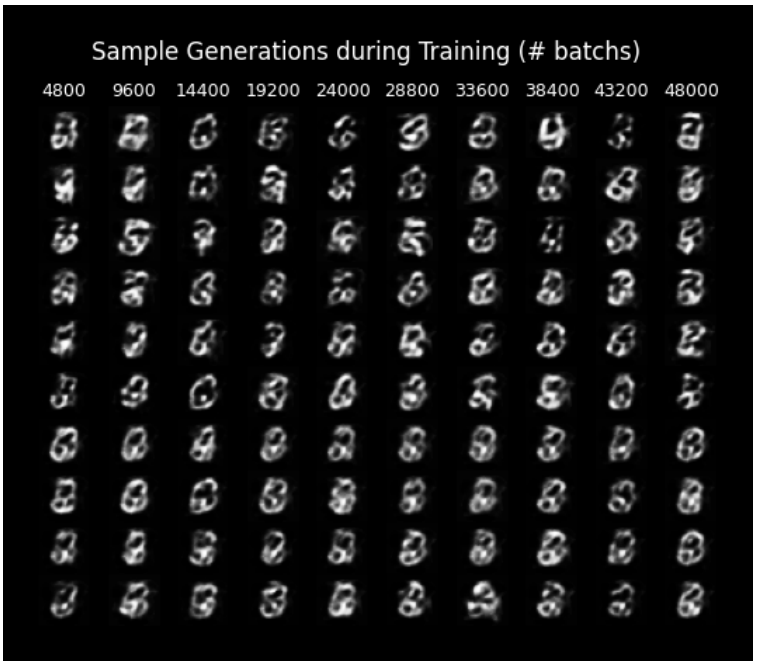

Figure 9: Generated images during the training of the ae-grad-VAE model. $\epsilon = 367.7$.

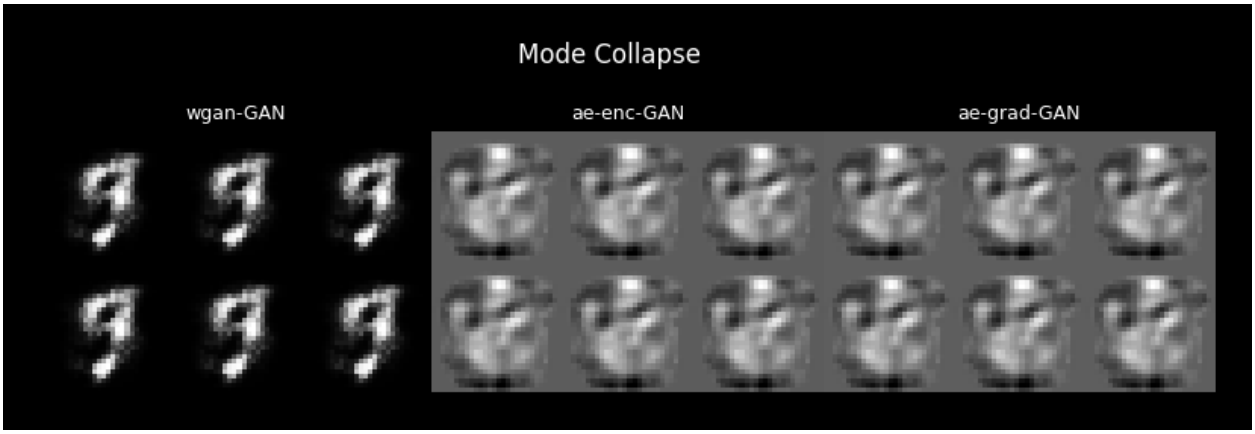

Figure 10: Reconstructions from mode collapsed models.