Differentially Private Latent Diffusion Models

Saiyue Lyu *
University of British Columbia
saiyuel@cs.ubc.ca

Margarita Vinaroz *
University of Tübingen
margarita.vinaroz@tuebingen.mpg.de

Michael F. Liu *
University of British Columbia
mfliu@cs.ubc.ca

Mijung Park
University of British Columbia
mijungp@cs.ubc.ca

Abstract

Diffusion models (DMs) are widely used for generating high-quality image datasets. However, since they operate directly in the high-dimensional pixel space, optimization of DMs is computationally expensive, requiring long training times. This contributes to large amounts of noise being injected into the differentially private learning process, due to the composability property of differential privacy. To address this challenge, we propose training *Latent* Diffusion Models (LDMs) with differential privacy. LDMs use powerful pre-trained autoencoders to reduce the high-dimensional pixel space to a much lower-dimensional latent space, making training DMs more efficient and fast. Unlike [Ghalebikesabi et al., 2023] that pre-trains DMs with public data then fine-tunes them with private data, we fine-tune only the *attention* modules of LDMs at varying layers with privacy-sensitive data, reducing the number of trainable parameters by approximately 96% compared to fine-tuning the entire DM. We test our algorithm on several public-private data pairs, such as ImageNet as public data and CIFAR10 and CelebA as private data, and SVHN as public data and MNIST as private data. Our approach provides a promising direction for training more powerful, yet training-efficient differentially private DMs that can produce high-quality synthetic images.

1 Introduction

Creating impactful machine learning solutions for real-world applications often requires access to personal data that may compromise privacy, raising ethical and legal concerns. Due to these reasons, *differentially private data generation* has become an active area of research. The main objective of this research is to generate synthetic data that preserves the privacy of the individuals in the original data while maintaining the statistical properties of the original data. Unlike traditional methods that require strict assumptions about the intended use of synthetic data [Mohammed et al., 2011, Xiao et al., 2010, Hardt et al., 2012, Zhu et al., 2017], recent approaches aim to create synthetic data that is general-purpose and useful for a range of downstream tasks, including training a classifier and performing statistical testing. These popular approaches include GAN-based models [Torkzadehmahani et al., 2019, Yoon et al., 2019, Chen et al., 2020], optimal transport or kernel-based distance approaches [Cao et al., 2021, Harder et al., 2021, Vinaroz et al., 2022, Yang et al., 2023], and diffusion models [Dockhorn et al., 2023].

Many of these popular approaches for differentially private data generation operate on small generative models, such as two-layer convolutional neural networks (CNNs), and simple datasets such as MNIST and FashionMNIST. This is because the DP training algorithm, called *differentially private stochastic*

*Equal contribution

Preprint. Under review.

gradient descent (DP-SGD)[Abadi et al., 2016], does not scale well for large models that are necessary for learning complex distributions. The recent approach by Dockhorn et al. [2023] uses diffusion models that have shown impressive performance on high-dimensional image generation in non-DP settings. However, due to the scalability issue of DP-SGD, DP trained diffusion models yield rather underwhelming performance when evaluated on complex datasets such as CIFAR10 and CelebA.

More recent approaches attempt to overcome this issue by utilizing the abundant resource of *public* data. For example, Harder et al. [2023] use public data for pre-training a large feature extractor model to learn useful features without incurring a privacy loss, then uses those features to train a generator using private data. Ghalebikesabi et al. [2023] pretrains a large diffusion-based generator using public data then fine-tunes it for private data for a relatively small number of epochs using DP-SGD. This method achieves the state-of-the-art performance on CIFAR10 image generation with differential privacy.

In this paper, we attempt to further improve the performance of differentially private image generation by reducing the number of parameters to fine-tune. To achieve this, we build off of *latent diffusion models (LDMs)* by Rombach et al. [2022], which uses a pre-trained autoencoder to reduce the size of images, often called *latent variables*, entering into the diffusion model. This latent diffusion model defined on this latent space has a significantly lower number of parameters than the diffusion model defined on the pixel space. Inspired by [You and Zhao, 2023] that establishes a transfer learning paradigm for LDMs in non-DP settings, we pre-train the entire LDM including the auto-encoder using public data, and fine-tune only attention modules and a conditioning embedder using our private data. As a result, the number of trainable parameters under our approach is only $3 - 5\%$ of that of the diffusion models used in [Ghalebikesabi et al., 2023] and achieves similar performance.

The use of DP-SGD to fine-tune a large model in this paper may appear unremarkable to readers, as it is a common approach. However, there is scarce literature on fine-tuning with differential privacy for image generation, a significantly more challenging task compared to a classification task. Only one previous work (not yet published) has explored fine-tuning diffusion models in this context [Ghalebikesabi et al., 2023], and there are no existing studies on fine-tuning LDMs. Given the unique aspects of our proposed method (training DMs in the latent space and fine-tuning lower-dimensional parameters compared to fine-tuning the entire network), our training process is considerably more efficient than training a DM from scratch with DP. Specifically, we can achieve similar performance while reducing the training time from 192 GPU hours (using eight V100 NVIDIA GPUs for approximately 24 hours) as reported in the work by Dockhorn et al. [2023], to just 3 GPU hours (using a single V100 NVIDIA GPU for 3 hours).

In the following section, we provide relevant background information. We then present our method along with related work and experiments on benchmark datasets.

2 Background

We first describe latent diffusion models (LDMs), then the definition of differential privacy (DP) and finally the DP-SGD algorithm, which we will use to train the LDM in our method.

2.1 Latent Diffusion Models (LDMs)

[Rombach et al., 2022] develop latent diffusion models (LDMs) by modifying denoising diffusion probabilistic models (DDPMs) Ho et al. [2020] in the following way. First, LDMs contain an auto-encoder, consisting of an encoder Δ and a decoder ∇, that transforms the high-dimensional pixel representations $\mathbf{x}$ into lower-dimensional latent representations $\mathbf{z}$ via $\mathbf{z} = \Delta(\mathbf{x})$, which are deemed to carry equivalent information as the former. The auto-encoder is trained with a combination of a perceptual loss and a patch-based adversarial objective, with extra regularization for better controlled variance in the learned latent space (See section 3 in [Rombach et al., 2022] for details). Training data transformed by the auto-encoder is lower-dimensional but contains encoded spatial structure of the pixel representations. Most importantly, this lower-dimensional data can expedite the training process of diffusion models, offering significantly reduced computational complexity (from hundreds of GPU days to several GPU hours for similar accuracy in non-DP settings).

Second, LDMs also contain *attention modules* that take inputs from a *conditioning embedder*, inserted into the layers of the underlying UNet backbone to achieve flexible conditional image generation (e.g.,

generating images conditioning on text, image layout, class labels, etc.). The modified Unet is then used as a function approximator $\tau_{\boldsymbol{\theta}}$ to predict an initial noise from the noisy lower-dimensional latent representations at several finite time steps t, where in LDMs, the noisy representations (rather than data) follow the diffusion process defined in Ho et al. [2020]. The parameters of the approximator are denoted by $\boldsymbol{\theta} = [\boldsymbol{\theta}^U, \boldsymbol{\theta}^{Att}, \boldsymbol{\theta}^{Cn}]$, where $\boldsymbol{\theta}^U$ are the parameters of the underlying Unet backbone, $\boldsymbol{\theta}^{Att}$ are the parameters of the attention modules, and $\boldsymbol{\theta}^{Cn}$ are the parameters of the conditioning embedder (We will explain these further in Subsec. 3.3). These parameters are then optimized by minimizing the prediction error defined by

$$\mathcal{L}_{ldm}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{z}_t, y), \tau, t} \left[\|\tau - \tau_{\boldsymbol{\theta}}(\mathbf{z}_t, t, y)\|_2^2 \right], \tag{1}$$

where $\tau \sim \mathcal{N}(0, I), \mathbf{z}_t = \Delta(\mathbf{x}_t)$ and y is what the model is conditioning on to generate data, e.g., class labels, or a prompt. Once the approximator is trained, the drawn samples in latent space, $\tilde{\mathbf{z}}$, are transformed back to pixel space through the decoder, i.e., $\tilde{\mathbf{x}} = \nabla(\tilde{\mathbf{z}})$.

Our work introduced in Sec. 3 pre-trains both the auto-encoder and the function approximator using public data, then fine-tunes only $\boldsymbol{\theta}_{Att}, \boldsymbol{\theta}_{Cn}$, the parameters of the attention modules and the conditioning embedder, using DP-SGD for private data.

2.2 Differential Privacy (DP)

A mechanism $\mathcal{M}$ is (ϵ, δ)-DP for a given $\epsilon \geq 0$ and $\delta \geq 0$ if and only if

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

for all possible sets of the mechanism's outputs S and all neighbouring datasets $\mathcal{D}$, $\mathcal{D}'$ that differ by a single entry. A single entry difference could come from either replacing or removing one entry from the dataset $\mathcal{D}$. One of the most well-known and widely used DP mechanisms is the *Gaussian mechanism*. The Gaussian mechanism adds a calibrated level of noise to a function $\mu : \mathcal{D} \mapsto \mathbb{R}^p$ to ensure that the output of the mechanism is (ϵ, δ)-DP: $\widetilde{\mu}(\mathcal{D}) = \mu(\mathcal{D}) + n$, where $n \sim \mathcal{N}(0, \sigma^2 \Delta_{\mu}^2 \mathbf{I}_p)$. Here, σ is often called a privacy parameter, which is a function of ϵ and δ. For instance, one could use the value of the privacy parameter that satisfies the theoretical bound given by $\sigma \geq \sqrt{2\log(1.25/\delta)}/\epsilon$, when $0 < \epsilon < 1$, to provide (ϵ, δ)-DP guarantee. A tighter bound can be achieved by numerically computing the bound using, e.g., `auto-dp` package [Wang et al., 2019], among others. Δ_{μ} is often called the *global sensitivity* [Dwork et al., 2006], which is the maximum difference in L_2-norm given two neighbouring $\mathcal{D}$ and $\mathcal{D}'$, $||\mu(\mathcal{D}) - \mu(\mathcal{D}')||_2$. Because we are adding noise, the natural consequence is that the released function $\tilde{\mu}(\mathcal{D})$ is less accurate than the non-DP counterpart, $\mu(\mathcal{D})$. This introduces privacy-accuracy trade-offs.

There are two important properties of DP. The *post-processing invariance* property of DP Dwork et al. [2006] implies that the composition of any data-independent mapping with an (ϵ, δ)-DP algorithm is also (ϵ, δ)-DP. So no analysis of the released synthetic data can yield more information about the real data than what our choice of ϵ and δ allows. The *composability* theorem Dwork et al. [2006] states that the strength of privacy guarantee degrades in a measurable way with repeated use of DP-algorithms. This composability property of DP poses a significant challenge in deep learning.

DP-SGD [Abadi et al., 2016] is one of the most widely used differentially private algorithms for training deep neural network models. It modifies stochastic gradient descent (SGD) by adding an appropriate amount of noise by employing the Gaussian mechanism to the gradients in every training step to ensure the parameters of a neural network are differentially private. However, there are two challenges in DP-SGD. First, it is infeasible to obtain an analytic sensitivity of gradients under complex deep neural network models. A remedy to this issue is explicitly normalizing the norm of each sample-wise gradient with some pre-chosen value C such that the gradient norm given any datapoint's difference between two neighbouring datasets cannot exceed C. Second, due to the composability property of DP, privacy loss is accumulating over a typically long course of training. Abadi et al. [2016] exploit the subsampled Gaussian mechanism (i.e., applying the Gaussian mechanism on randomly subsampled data) to achieve a tight privacy bound. The *Opacus* package [Yousefpour et al., 2021] implements the DP-SGD algorithm, which we adopt in our method.

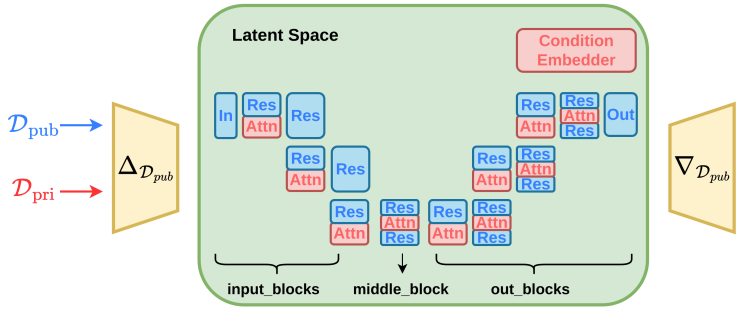

Figure 1: A schematic of DP-LDM. In the first step, we pre-train the auto-encoder depicted in yellow (Right and Left) with public data. In the second step, we forward pass the public data through the encoder (blue arrow on the left) to obtain latent representations. We then train the diffusion model (depicted in the green box) on the lower-dimensional latent representations. The diffusion model consists of the Unet backbone and added attention modules (in Red) with a conditioning embedder (in Red, at top-right corner). In the third step, we forward pass the private data (red arrow on the left) through the encoder to obtain latent representations of the private data. We then fine-tune only the red blocks, which are attention modules and conditioning embedder, with DP-SGD. Once the training is done, we sample the latent representations from the diffusion model, and pass them through the decoder to obtain the image samples in the pixel space.

3 Differentially private latent diffusion models (DP-LDMs)

In our method, which we call *differentially private latent diffusion models (DP-LDMs)*, we carry out three training steps. In what follows, we describe each of these steps, with an emphasis on the last step, the DP fine-tuning step, in more detail.

3.1 Pre-training an autoencoder using public data

Following Rombach et al. [2022], we first pre-train an auto-encoder. The encoder scales down an image $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ to a 3-dimensional latent representation $\mathbf{z} \in \mathbb{R}^{h\times w\times c}$ by a factor of f, where $f = H/h = W/w$. This 3-dimensional latent representation is chosen to take advantage of image-specific inductive biases that the Unet contains, e.g., 2D convolutional layers. Following Rombach et al. [2022], we also train the autoencoder by minimizing a combination of different losses, such as perceptual loss and adversarial loss, with some form of regularization. See Appendix for details. As noted by Rombach et al. [2022], we also observe that a mild form of downsampling performs the best, achieving a good balance between training efficiency and perceptually decent results. See Appendix for details on different scaling factors $f = 2^m$, with $m \in \{1, 2, 4, 8\}$.

This step of training an auto-encoder does not incur any privacy loss, as we use public data $\mathcal{D}_{pub}$ that is similar to private data $\mathcal{D}_{priv}$ at hand. The trained autoencoder is, therefore, a function of public data: an encoder $\Delta_{\mathcal{D}_{pub}}$ and a decoder $\nabla_{\mathcal{D}_{pub}}$.

3.2 Pre-training a diffusion model with latent representations of public data

A forward pass through the trained encoder $\Delta_{\mathcal{D}_{pub}}$ gives us a latent representation of each image, which we use to train a diffusion model. As in [Rombach et al., 2022], we also consider a modified Unet for the function approximator $\tau_{\boldsymbol{\theta}}$, as shown in Fig. 1. Recall that there are three sets of parameters of the approximator: $\boldsymbol{\theta} = [\boldsymbol{\theta}^U, \boldsymbol{\theta}^{Att}, \boldsymbol{\theta}^{Cn}]$, where $\boldsymbol{\theta}^U$ are the parameters of the underlying Unet backbone, $\boldsymbol{\theta}^{Att}$ and $\boldsymbol{\theta}^{Cn}$ are the parameters of the attention modules and a conditioning embedder, respectively. We minimize the loss to estimate the parameters of $\tau_{\boldsymbol{\theta}}$ as:

$$\boldsymbol{\theta}^U_{\mathcal{D}_{pub}}, \boldsymbol{\theta}^{Att}_{\mathcal{D}_{pub}}, \boldsymbol{\theta}^{Cn}_{\mathcal{D}_{pub}} = \arg\min_{\boldsymbol{\theta}} \ \mathcal{L}_{ldm}(\boldsymbol{\theta}), \tag{2}$$

where the loss function is given in eq. 1. since we are using public data, we also do not incur any privacy loss in this step. These parameters in our model are a function of public data $\mathcal{D}_{pub}$.

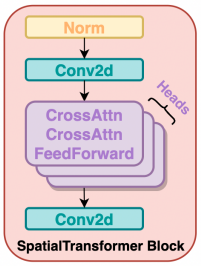

Figure 2: A SpatialTransformer Block

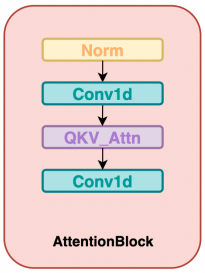

Figure 3: An AttentionBlock

3.3 Fine-tuning attention modules & conditioning embedder for private data

Given a pre-trained diffusion model, we fine-tune the attention modules and a conditioning embedder using our private data. That is, we fix the Unet backbone parameters to $\boldsymbol{\theta}^{U}_{\mathcal{D}_{pub}}$, while training $\boldsymbol{\theta}^{Att}$ and $\boldsymbol{\theta}^{Cn}$ starting from the values of $\boldsymbol{\theta}^{Att}_{\mathcal{D}_{pub}}, \boldsymbol{\theta}^{Cn}_{\mathcal{D}_{pub}}$.

For the models with the conditioned generation, the attention modules refer to the spatial transformer blocks shown in Fig. 2 which contains two cross attention and can have multiple heads. For the models with an unconditional generation, the attention modules refer to the attention blocks shown in Fig. 3. Consequently, the parameters of the attention modules, denoted by $\boldsymbol{\theta}^{Att}$, differ, depending on the conditioned or unconditioned cases. The conditioning embedder only exists in the conditioned case. Depending on the different modalities the model is trained on, the conditioning embedder takes a different form. For instance, if the model generates images conditioning on the class labels, the conditioning embedder is simply a class embedder, which embeds class labels to a latent dimension. If the model conditions on language prompts, the embedder can be a transformer.

The core part of the spatial transformer block and the attention block is the attention layer, which has the following parameterization (For simplicity, we explain it under the conditioned case):

$$\text{Attention}(\psi_i(\mathbf{z}_t), \phi(y); Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \ \in \mathbb{R}^{N\times d_k}, \tag{3}$$

where $\psi_i(\mathbf{z}_t) \in \mathbb{R}^{N\times d^i}$ is an intermediate representation of the latent representation $\mathbf{z}_t$ through the ith residual convolutional block in the backbone Unet, and $\phi(y) \in \mathbb{R}^{M\times d_c}$ is the embedding of what the generation is conditioned on (e.g., class labels, CLIP embedding).

$$Q = \psi_i(\mathbf{z}_t) W_Q^{(i)\top} \ \in \mathbb{R}^{N\times d_k} \tag{4}$$

$$K = \phi(\mathbf{x}) W_K^{(i)\top} \ \in \mathbb{R}^{M\times d_k} \tag{5}$$

$$V = \phi(\mathbf{x}) W_V^{(i)\top} \ \in \mathbb{R}^{M\times d_k}, \tag{6}$$

where the parameters are denoted by $W_Q^{(i)} \in \mathbb{R}^{d_k\times d^i}$; $W_K^{(i)} \in \mathbb{R}^{d_k\times d_c}$; and $W_V^{(i)} \in \mathbb{R}^{d_k\times d_c}$. Unlike the conditioned case, where the key (K) and value (V) vectors are computed as a projection of the conditioning embedder, the key and value vectors are a projection of the pixel embedding $\psi_i(\mathbf{z}_t)$ only in the case of the unconditioned model. We run DP-SGD to fine-tune these parameters to obtain differentially private $\boldsymbol{\theta}^{Att}_{\mathcal{D}_{priv}}$ and $\boldsymbol{\theta}^{Cn}_{\mathcal{D}_{priv}}$. Our algorithm is given in Algorithm 1.

Why does fine-tuning the attention modules and a conditioning embedder make sense? The output of the attention in eq. 3, a filtered value matrix, assigns a high focus to the features that are more important, by zooming into what truly matters in an image depending on a particular context, e.g., relevant to what the image is conditioned on. This can be quite different when we move from a public dataset to a private dataset. Hence, we fine-tune the attention modules (together with the conditioning embedder when conditioned case), which effectively transfers what we learned from the public data distribution to the private data distribution, as shown in Sec. 5. A similar idea is also explored in the context of language modelling [Yu et al., 2022], which shows the effectiveness of fine-tuning the Transformer attention weights (reparameterized to be parameter efficient) at the scale of large language models such as GPT-3 and GPT-2-XL. However, this paper proposes to fine-tune

Algorithm 1 DP-LDM

Input: Latent representations through a pre-trained auto-encoder and conditions (if conditioned generation) $\{(\mathbf{z}_i, y_i)\}_{i=1}^N$, a pre-trained diffusion model with parameters $\boldsymbol{\theta}$, number of iterations P, mini-batch size B, clipping norm C, learning rate η, privacy parameter σ corresponding to (ϵ, δ)-DP. Denote $\hat{\boldsymbol{\theta}} = \{\boldsymbol{\theta}^{Att}, \boldsymbol{\theta}^{Cn}\}$

for $p = 1$ **to** P **do**

Step 1. Take a mini-batch B_p uniformly at random with a sampling probability, $q = B/N$

Step 2. For each sample $i \in B_p$ compute the gradient: $g_p(\mathbf{z}_i, y_i) = \nabla_{\hat{\boldsymbol{\theta}}_p} \mathcal{L}_{ldm}(\hat{\boldsymbol{\theta}}_p, \mathbf{z}_i, y_i)$

Step 3. Clip the gradient: $\hat{g}_p(\mathbf{z}_i, y_i) = g_p(\mathbf{z}_i, y_i) / \max(1, \|g_p(\mathbf{z}_i, y_i)\|_2 / C)$

Step 4. Add noise: $\tilde{g}_p = \frac{1}{B} \sum_{i=1}^{B} \hat{g}_p(\mathbf{z}_i, y_i) + \mathcal{N}(0, \sigma^2 C^2 I)$

Step 5. Gradient descent: $\hat{\boldsymbol{\theta}}_{p+1} = \hat{\boldsymbol{\theta}}_p - \eta \tilde{g}_p$

end for

Return: (ϵ, δ)-differentially private $\hat{\boldsymbol{\theta}}_P = \{\boldsymbol{\theta}_P^{Att}, \boldsymbol{\theta}_P^{Cn}\}$

the attention modules in the Unet model for image generation, which has not been studied with differential privacy.

4 Related Work

Early efforts in differentially private data generation imposes strict limitations on the data type and the intended purpose of the released data [Snoke and Slavković, 2018, Mohammed et al., 2011, Xiao et al., 2010, Hardt et al., 2012, Zhu et al., 2017], which leads to the difficulties in generating large-scale data. Later, several works have explored generating discrete data with restricted range of values, by understanding the relationships of small groups of features and then privatizing them [Zhang et al., 2017, Qardaji et al., 2014, Chen et al., 2015, Zhang et al., 2021]. However, these techniques cannot be applied to high-dimensional data due to the constraint of discretization. Recently, more efforts have focused on leveraging advanced generative models to achieve better differentially private synthetic data. Some of them [Xie et al., 2018, Torkzadehmahani et al., 2019, Frigerio et al., 2019, Yoon et al., 2019, Chen et al., 2020] utilize generative adversarial networks (GANS)[Goodfellow et al., 2014], or trained GANs with the PATE structure [Papernot et al., 2017]. Other works have employed variational autoencoders (VAEs) [Acs et al., 2018, Jiang et al., 2022, Pfitzner and Arnrich, 2022], or proposed customized structures [Harder et al., 2021, Vinaroz et al., 2022, Cao et al., 2021, Liew et al., 2022, Harder et al., 2023]. For instance, [Harder et al., 2023] pretrained perceptual features using public data and privatized only data-dependent terms using maximum mean discrepancy.

Limited works have so far delved into differentially private stochastic gradient descent (DP-SGD) [Abadi et al., 2016] on diffusion models. Dockhorn et al. [2023] build on score-based generative models Song et al. [2021] and enforced privacy with DP-SGD. However, they only focused on simple datasets such as MNIST, FashionMNIST and CelebA (downsampled to 32×32). In addition, it takes eight NVIDIA V100 GPUs training for roughly one day to achieve good downstream accuracy. Our method has efficiently reduced the training expense to 3 hours using a single NVIDIA V100 GPU while achieving similar results. Ghalebikesabi et al. [2023] recently fine-tuned the ImageNet pre-trained diffusion model (DDPM) [Ho et al., 2020] with more than 80 M parameters using DP-SGD on CIFAR-10, which is a more complex high-dimensional dataset. However, their DP fine-tuning is not as straightforward as ours, since a modified time-step scheduling (different from the small time steps at which the pre-trained model was trained) is required to fine-tune the model with the extra noise due to differential privacy constraints.

Inspired by the Latent Diffusion Model (LDM) [Rombach et al., 2022], we propose to fine-tune the pretrained LDM in the latent space to significantly reduce the number of trainable parameters, which will benefit the performance of DP-SGD. Following [You and Zhao, 2023], we considered only fine-tuning the attention modules and the conditioning embedder in the model to further reduce the model size while achieving comparable FID scores with quicker training. In this paper, we demonstrate that our approach is capable of dealing with both simple and complex datasets for differentially private data generation.

5 Experiments

Here, we demonstrate the performance of our method in comparison with the state-of-the-art methods in DP data generation, using several combinations of public/private data of different levels of complexity at varying privacy levels.

Datasets. We considered four image datasets[2] of varying complexity. We started with the commonly used datasets MNIST [LeCun and Cortes, 2010] and FashionMNIST [Xiao et al., 2017] (See Appendix for FashionMNIST results), where each consist of 60,000 28×28 pixel grayscale images depicting hand-written digits and i our experimentstems of clothing, respectively, sorted into 10 classes. We also looked at the more complex CelebA [Liu et al., 2015] dataset, containing 202,599 color images of faces which we scale to sizes of 32×32 or 64×64 pixels (See Appendix for the results of 64×64 pixels generation) and treat as unlabeled. We also study CIFAR-10 [Krizhevsky et al., 2009], a 50,000-sample dataset containing 32×32 color images of 10 classes of objects, including vehicles like ships and trucks, and animals such as horses and birds.

5.1 Transferring from SVHN to MNIST distribution

We use SVHN as public data to pretrain the conditional LDM with SpatialTransformer Block in Fig. 2. We compare DP-LDM to existing methods on the most common DP settings with $\epsilon = 10, \delta = 10^{-5}$ in Table 1. Our results outperform SOTA results except for [Dockhorn et al., 2023], however, our model uses much less training time with significantly smaller batch size on only a single GPU. We think our method may achieve much better performance with a larger memory, as it is known that using larger batch sizes is beneficial in improving the signal-to-noise ratio in DP-SGD training. See Appendix for results on FashionMNIST.

We also did experiments for ablation in Table 2. By reducing the number of trainable attention modules, we find there is a slight performance decrease and an increase in the standard deviation. Moreover, only tuning 5-7 modules still gives comparable downstream accuracies, which means the output_blocks is dominating the training. These ablation results provide a convincing direction for further reducing tuning parameters. Details for the number of parameters are in the Appendix.

Table 1: Downstream accuracies by Logistic regression, MLP and CNN, evaluated on the generated data samples using MNIST and FashionMNIST as private data and SVHN and CIFAR-10 as public data, respectively. In all cases, we set $\epsilon = 10, \delta = 10^{-5}$. Note that Ghalebikesabi et al. [2023] achieved 98.6% accuracy on test set of MNIST, while their classifier was significantly larger (WRN-40-4) than the classifiers below. We ran the experiments three times with different seeds and reported the mean and the std.

		DP-LDM (Ours)	DP-MEPF [Harder et al., 2023]	DP-Sinkhorn [Cao et al., 2021]	GS-WGAN [Chen et al., 2020]	DP-HP [Vinaroz et al., 2022]	DP-DM [Dockhorn et al., 2023]
MNIST	CNN	94.25 ± 0.02	NA	83.2	80	NA	98.1
	MLP	90.96 ± 0.04	90	83	79	82	94.8
	LogReg	85.02 ± 0.01	83	83	79	81	90.8
	GPU	1 V100	NA	NA	NA	NA	8 V100
	Hours	3 hour	NA	NA	NA	NA	1 day
	BatchSize	256	NA	NA	NA	NA	4096

Table 2: Downstream accuracies for ablation for $\epsilon = 10, \delta = 10^{-5}$. The Unet we used for MNIST has 7 attention modules, 1-2 are in input_blocks, 3 is in middle block, 4-7 are in output_blocks. We calculated accuracies for fine tuning 1-7 (values in Table 1), fine tuning 3-7, and fine tuning 5-7.

		1-7(all)	3-7	5-7
MNIST	CNN	94.25 ± 0.02	93.29 ± 0.36	92.34 ± 1.15
	MLP	90.96 ± 0.04	88.83 ± 0.91	85.71 ± 2.30
	LogReg	85.02 ± 0.01	83.93 ± 2.36	82.95 ± 0.35

[2]Dataset licenses: MNIST: CC BY-SA 3.0; FashionMNIST:MIT; CelebA: see `https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`; Cifar10: MIT

5.2 Transferring from Imagenet to CIFAR10 distribution

First, we pre-trained a class conditional LDM model considering ImageNet32 as public data. The Unet we use has 16 SpatialTransformer blocks as in Fig. 2 For the fine-tuning part, we consider CIFAR-10 as the private dataset and test the performance of DP-LDM at different privacy levels for 3 independent sampling runs. See Appendix for details.

We start by conducting ablation experiments to test the performance of DP-LDM when fine-tuning only certain attention modules inside the pre-trained model and keeping the rest of the parameters fixed. Table 3 shows the FID obtained for $\epsilon = 1, 5, 10$ and $\delta = 10^{-5}$ for the different number of attention modules fine-tuned. The results show that fixing up to the first half of the attention layers in the LDM has a positive effect in terms of the FID (the lower the better) in our model.

The intuition behind this behaviour is that reducing the number of trainable parameters is beneficial at a high privacy regime (i.e., smaller ϵ). However, fixing most of the layers does not give the best results. As illustrated in Table 3, fine-tuning with 9-16 will achieve the highest scores. The comparison to other methods in terms of FID is given in Table 4. We suspect that the large difference in FID between ours and [Ghalebikesabi et al., 2023] is due to the batch size. Their batch size was 16,384, while the largest batch size we could fit in our memory was about 200. Our generated images are in Appendix. We also compare the performance of DP-LDM in terms of downstream classification accuracy based on ResNet-9 with other current SOTA differentially private generative methods in Appendix.

Table 3: FID scores (lower is better) for synthetic CIFAR-10 data with varying the number of fine-tuning layers and privacy guarantees. **Top row (1-16 layers):** Fine-tuning all attention modules. **Second row (5-16 layers):** Keep first 4 attention modules fixed and fine-tuning from 5 to 16 attention modules. **Third row (9-16 layers):** Keep first 8 attention modules fixed and fine-tuning from 9 to 16 attention modules. **Bottom row (13-16 layers):** Keep first 12 attention modules fixed and fine-tuning from 13 to 16 attention modules.

DP-LDM	$\epsilon = 10$	$\epsilon = 5$	$\epsilon = 1$
1-16 layers	35.1 ± 0.2	36.5 ± 0.3	37.6 ± 0.1
5 - 16 layers	30.7 ± 0.2	31.8 ± 0.2	33.4 ± 0.4
9 - 16 layers	19.2 ± 0.2	23.3 ± 0.1	25.2 ± 0.2
13 - 16 layers	27.8 ± 0.2	28.7 ± 0.4	30.8 ± 0.3

Table 4: FID scores for synthetic CIFAR-10 data. Values for [Ghalebikesabi et al., 2023] and DP-MEPF are the ones reported in their papers.

	$\epsilon = 10$	$\epsilon = 5$	$\epsilon = 1$
DP-LDM	19.2	23.3	25.2
[Ghalebikesabi et al., 2023]	9.8	15.1	25.2
DP-MEPF (ϕ_1, ϕ_2)	26.6	27.6	38.6
DP-MEPF (ϕ_1)	27.1	27.7	39.0

5.3 Transferring from Imagenet to CelebA distribution

We evaluate the performance of our model on the task of unconditional image generation for CelebA. We use the same autoencoder as in section 5.2, but pretrain a new LDM on ImageNet32 without class labels. Since there is no conditioning context, the SpatialTransformer blocks (Figure 2) are replaced with AttentionBlocks (Figure 3). Since this is unconditional case, we do not include the conditioning embedder in this experiment.

The LDM is fine-tuned with differential privacy on CelebA images rescaled and cropped to 32x32 resolution for $\epsilon \in \{1, 10\}$ and $\delta = 10^{-5}$. FID results are shown in Table 5. We discovered an error in the FID computation for DP-MEPF taken from [Harder et al., 2023], and have contacted the authors[3]. As such, we primarily consider DP-GAN, DPDM, and DP-Sinkhorn. Since our privacy parameter $\delta = 10^{-5}$, results shown in Figure 5 are note directly comparable. See Appendix for our FID results with $\delta = 10^{-6}$ and experiments with CelebA 64x64.

[3]See the disclaimer by the authors: `https://github.com/ParkLabML/DP-MEPF/tree/main`

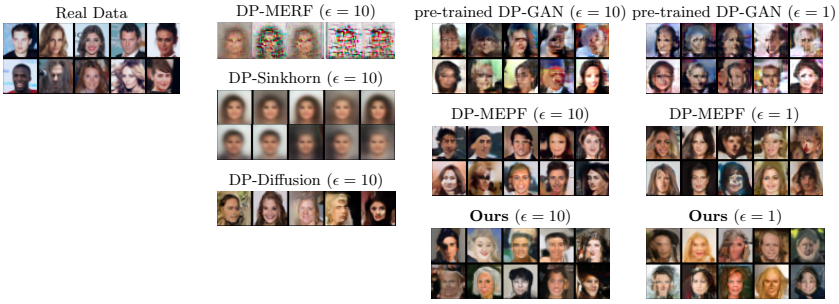

Figure 4: Synthetic 32×32 CelebA samples generated at different levels of privacy. Samples for DP-MERF and DP-Sinkhorn are taken from [Cao et al., 2021], DP-Diffusion samples are taken from [Dockhorn et al., 2022], and DP-MEPF samples are taken from [Harder et al., 2023].

Table 5: CelebA FID scores (lower is better) for images of resolution 32×32. Results for DP Diffusion (DPDM), DP Sinkhorn, and DP MEPF taken from [Dockhorn et al., 2023], [Cao et al., 2021], and [Harder et al., 2023]. Note that there is an error in the calculation of FID for DP-MEPF

	δ	$\epsilon = 10$	$\epsilon = 1$
DP-LDM (ours)	10^{-5}	21.0 ± 0.6	27.4 ± 0.4
DP-MEPF (ϕ_1)	10^{-6}	~~11.7~~	~~13.2~~
DP-GAN (pre-trained)	10^{-6}	40.3	53.3
DPDM (no public data)	10^{-6}	21.2	71.8
DP Sinkhorn (no public data)	10^{-6}	189.5	-

6 Conclusion and Discussion

We proposed *Differentially Private Latent Diffusion Models* (DP-LDM), which utilized DP-SGD to fine tune the attention modules of the pretrained LDM at varying layers with privacy-sensitive data. We demonstrate that our method takes advantage of using auxiliary public data in DP data generation, and is capable of tackling both simple datasets like MNIST and complex high-dimensional datasets like CIFAR-10 and CelebA under private settings. We perform an in-depth analysis of ablation of DP-LDM to explore the strategy to reducing parameters for more applicable training of DP-SGD. Based on our promising results, we conclude that fine tuning LDMs is an efficient and effective framework for DP generative learning. We hope our results can contribute to future research on DP data generation considering the rapid advances in diffusion model-based image generation. In addition, given the difficulties of training with DP-SGD, we hope our framework can lead to faster and more efficient training with a small amount of computing resources by analytically reducing the number of trainable parameters.

7 Societal Impact

As investigated in [Carlini et al., 2023], diffusion models can memorize individual images from the training data and gives the same as generating samples. Aiming to bringing positive effects to society, our research is driven by the necessity of robust and scalabel data privacy. However, it is also important to approach the use of public data cautiously. As [Tramèr et al., 2022] pointed out, public data themselves may still be sensitive due to lack of curation practices. In addition, the public data usually achieve similar distribution as the private data, however, no proper public data is available currently as this might require heavy domain shift of the data themselves. We understand these potential issues but current DP machine learning research leads to minimal effects because of the inadequacy of the utility. From our perspective, auxiliary public data still emerges as the most promising option for attaining satisfactory utility, comparing to the potential harm it might inject. We hope our discussion will contribute to further research in differential privacy in machine learning using public data.

Acknowledgements

All the authors are supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada CIFAR AI Chairs program. We are grateful for the computational resources given by Advanced Research Computing at the University of British Columbia and the Digital Research Alliance of Canada. We thank You and Zhao [2023] for sharing their preliminary code.

References

Martin Abadi, Andy Chu, Ian J. 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*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318.

Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1109–1121, 2018.

Tianshi Cao, Alex Bie, Arash Vahdat, Sanja Fidler, and Karsten Kreis. Don't generate me: Training differentially private generative models with sinkhorn divergence. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. In *Advances in Neural Information Processing Systems 33*, 2020.

Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138, 2015.

Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. Differentially private diffusion models. *arXiv preprint arXiv:2210.09929*, 2022.

Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. Differentially private diffusion models, 2023. URL `https://openreview.net/forum?id=pX21pH4CsNB`.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006. doi: 10.1007/11761679_29.

Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *ICT Systems Security and Privacy Protection - 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings*, pages 151–164, 2019. doi: 10.1007/978-3-030-22312-0_11.

Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L. Smith, Olivia Wiles, and Borja Balle. Differentially private diffusion models generate useful synthetic images, 2023.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2014.

Frederik Harder, Kamil Adamczewski, and Mijung Park. DP-MERF: Differentially private mean embeddings with random features for practical privacy-preserving data generation. In *AISTATS*, volume 130 of *Proceedings of Machine Learning Research*, pages 1819–1827. PMLR, 2021.

Frederik Harder, Milad Jalali, Danica J. Sutherland, and Mijung Park. Pre-trained perceptual features improve differentially private image generation. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=R6W7zkMz0P.

Moritz Hardt, Katrina Ligett, and Frank Mcsherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25*, pages 2339–2347. Curran Associates, Inc., 2012.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Dihong Jiang, Guojun Zhang, Mahdi Karami, Xi Chen, Yunfeng Shao, and Yaoliang Yu. Dp^2-vae: Differentially private pre-trained variational autoencoders. *arXiv preprint arXiv:2208.03409*, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, ON, Canada, 2009.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL http://yann.lecun.com/exdb/mnist/.

Seng Pei Liew, Tsubasa Takahashi, and Michihiko Ueno. PEARL: Data synthesis via private embeddings and adversarial reconstruction learning. In *International Conference on Learning Representations*, 2022.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Noman Mohammed, Rui Chen, Benjamin C.M. Fung, and Philip S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 493–501, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020487.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Bjarne Pfitzner and Bert Arnrich. Dpd-fvae: Synthetic data generation using federated variational autoencoders with differentially-private decoder. *arXiv preprint arXiv:2211.11591*, 2022.

Wahbeh Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

Joshua Snoke and Aleksandra Slavković. pmse mechanism: differentially private synthetic data with maximal distributional similarity. In *International Conference on Privacy in Statistical Databases*, pages 138–159. Springer, 2018.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2021.

Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. Dp-cgan: Differentially private synthetic data and label generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

Florian Tramèr, Gautam Kamath, and Nicholas Carlini. Considerations for differentially private learning with large-scale public pretraining. *arXiv preprint arXiv:2212.06470*, 2022.

Margarita Vinaroz, Mohammad-Amin Charusaie, Frederik Harder, Kamil Adamczewski, and Mi Jung Park. Hermite polynomial features for private data generation. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 22300–22324. PMLR, 2022.

Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *AISTATS*, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Secure Data Management*, pages 150–168, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15546-8.

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

Yilin Yang, Kamil Adamczewski, Danica J. Sutherland, Xiaoxiao Li, and Mijung Park. Differentially private neural tangent kernels for privacy-preserving data generation, 2023.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

Fuming You and Zhou Zhao. Transferring pretrained diffusion probabilistic models, 2023. URL `https://openreview.net/forum?id=8u9eXwu5GAb`.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Q42f0dfjECO`.

Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42 (4):1–41, 2017.

Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. Privsyn: Differentially private data synthesis. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

T. Zhu, G. Li, W. Zhou, and P. S. Yu. Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1619–1638, August 2017. ISSN 1041-4347. doi: 10.1109/TKDE.2017.2697856.

Appendix

A Implementation

Our code is available at `https://anonymous.4open.science/r/DP-LDM-4525`.

B Hyperparameters of Pretrained Models

Here we provide an overview of the hyperparameters of the pretrained autoencoder in Table 6, hyperparameters of the pretrained diffusion models in Table 7. Note that $base\ learning\ rate$ is the one set in the yaml files. The real learning rate passed to the optimizer is $accumulate\_grad\_batches \times num\_gpus \times batch\ size \times base\ learning\ rate$.

Table 6: Hyperparameters for the pretrained autoencoders for different datasets.

	SVHN (to MNIST)	CIFAR10 (to FMNIST)	Imagenet (to CIFAR10)	Imagenet (to CelebA 32)
Input size	32	32	32	32
Latent size	4	4	16	16
f	8	8	2	2
z-shape	$4 \times 4 \times 3$	$4 \times 4 \times 3$	$16 \times 16 \times 3$	$16 \times 16 \times 3$
Channels	128	128	128	128
Channel multiplier	[1,2,3,5]	[1,2,3,5]	[1,2]	[1,2]
Attention resolutions	[32,16,8]	[32,16,8]	[16, 8]	[16, 8]
num_res_blocks	2	2	2	2
Batch size	12	24	16	16
Base learning rate	4.5×10^{-6}	4.5×10^{-6}	4.5×10^{-6}	4.5×10^{-6}
Learning rate	1.08×10^{-4}	2.16×10^{-4}	1.4×10^{-4}	1.4×10^{-4}
Epochs	15	31	2	2
GPU(s)	1 NVIDIA V100	1 NVIDIA V100	1 NVIDIA RTX A4000	1 NVIDIA RTX A4000
Time	5 hours	10 hours	1 day	1 day

Table 7: Hyperparameters for the pretrained diffusion models for different datasets.

	SVHN (to MNIST)	CIFAR10 (to FMNIST)	Imagenet (to CIFAR10)	Imagenet (to CelebA 32)
input size	32	32	32	32
latent size	4	4	16	16
f	8	8	2	2
z-shape	$4 \times 4 \times 3$	$4 \times 4 \times 3$	$16 \times 16 \times 3$	$16 \times 16 \times 3$
channels	64	64	128	192
channel multiplier	[1,2]	[1,2]	[1,2,2,4]	[1, 2, 4]
attention resolutions	[1,2]	[1,2]	[1,2,4]	[1, 2, 4]
num_res_blocks	1	1	2	2
num_heads	4	4	8	-
num_head_channels	-	-	-	32
batch size	128	128	200	384
base learning rate	5×10^{-6}	5×10^{-6}	5×10^{-7}	5×10^{-7}
learning rate	6.4×10^{-4}	6.4×10^{-4}	2×10^{-4}	2×10^{-4}
epochs	60	77	30	13
# trainable parameters	4.6M	4.6M	90.8M	162.3M
GPU(s)	1 NVIDIA V100	1 NVIDIA V100	1 NVIDIA RTX A4000	1 NVIDIA V100
time	6 hours	7 hours	7 days	30 hours
use_spatial_transformer	True	True	True	False
cond_stage_key	class_label	class_label	class_label	-
conditioning_key	crossattn	crossattn	crossattn	-
num_classes	10	10	1000	-
embedding dimension	2	2	512	-
transformer depth	1	1	1	-

C Additional Experiment Results and Hyperparmeters

C.1 Transferring from SVHN to MNIST distribution

Here we attach more details for ablation experiments for $\epsilon = 10$ and $\delta = 10^{-5}$ for MNIST. Our UNet has 7 attention modules: 1-2 are in input_blocks, 3 is in the middle block, and 4-7 are in output_blocks. As discussed in [You and Zhao, 2023], tuning the input_blocks perform much worse than tuning the out_blocks. Therefore, we show experiments for tuning $a - 7$ blocks in Table 8, where $a \in \{3, 4, 5, 6, 7\}$. We see in the table that reducing the number of trained attention modules also leads to a decrease in accuracy.

Table 8: MNIST Downstream accuracies for ablation for $\epsilon = 10$, $\delta = 10^{-5}$ using seeds $\{21, 22, 23\}$.

		1-7	3-7	4-7	5-7	6-7	7
MNIST	CNN	94.25 ± 0.02	93.29 ± 0.36	92.57 ± 0.63	92.34 ± 1.15	90.18 ± 0.41	80.01 ± 0.60
	MLP	90.96 ± 0.04	88.83 ± 0.91	86.75 ± 1.01	85.71 ± 2.30	79.23 ± 0.87	68.76 ± 1.61
	LogReg	85.02 ± 0.01	83.93 ± 2.36	81.12 ± 1.18	82.95 ± 0.35	69.95 ± 1.31	53.98 ± 2.01
	N_{params}	1.6M(34.2%)	1.2M(25.2%)	828K(18%)	498K(10.82%)	167K(3.63%)	83K(1.81%)
	GPU(s)	1 V100	1 V100	1 V100	1 V100	1 V100	1 V100
	Time	3 hours	3 hours	3 hours	3 hours	3 hours	3 hours

We also include the experiment results for different epsilons with $\delta = 10^{-5}$ in Table 9

Table 9: MNIST Downstream accuracies different epsilons using seeds $\{21, 22, 23\}$.

		$\epsilon = \infty$	$\epsilon = 10$	$\epsilon = 1$
MNIST	CNN	98.45 ± 0.12	94.25 ± 0.02	88.82 ± 1.26
	MLP	95.33 ± 0.08	90.96 ± 0.04	81.57 ± 1.21
	LogReg	89.30 ± 0.01	85.02 ± 0.01	73.63 ± 0.95

Table 10 shows the hyperparameters we used for fine-tuning on MNIST and FMNIST:

Table 10: Hyperparameters for fine-tuning diffusion models with DP constraints $\epsilon = 10$ and $\delta = 10^{-5}$ on MNIST and FMNIST. The "ablation" hyperparameter determines which attention modules are fine-tuned, where a value of a means that the first $a - 1$ attention modules are frozen and others are trained. Setting "ablation" to -1 (default) fine-tunes all attention modules.

	MNIST (from SVHN)	FMNIST (from CIFAR10)
batch size	256	256
base learning rate	5×10^{-6}	5×10^{-6}
learning rate	1.28×10^{-3}	1.28×10^{-3}
epochs	60	80
num of params	1.6M	1.6M
use_spatial_transformer	True	True
cond_stage_key	class_label	class_label
conditioning_key	crossattn	crossattn
num_classes	10	10
embedding dimension	2	2
transformer depth	1	1
num of condition params	20	20
train_condition_only	True	True
attention_flag	spatial	spatial
epsilon	10	10
delta	10^{-5}	10^{-5}
clipping norm	0.1	0.01
noise scale	0.61	0.61
ablation	-1	-1

C.2 Transferring from CIFAR10 to FMNIST

Following [Harder et al., 2023], we also investigated the results of transferring from CIFAR10 as public dataset to FMNIST as private dataset. The results are illustrated in Table 11.

Table 11: FMNIST Downstream accuracies at a different value of epsilon using seed = 22.

		$\epsilon = \infty$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 0.2$
Ours (DP-LDM)	CNN	76	77	74	73
	MLP	75	73	70	70
	LogReg	70	65	62	60
[Harder et al., 2023]	CNN	-	-	-	-
	MLP	74	76	75	70
	LogReg	73	76	75	73

C.3 Transferring from Imagenet to CIFAR10 distribution

In this section, we first investigate DP-LDM performance on the downstream classification test accuracy based on ResNet9 for different ϵ values after three sampling independent runs. Table 12 shows the mean and standard deviation after three sampling independent runs for DP-LDM fine-tuned at different number of attention modules for CIFAR-10. In Table 13 we compare our best DP-LDM result for fine-tuning CIFAR-10 with current SOTA differentially private generative methods. We also report in Table 14 the different hyper-parameters used in DP-LDM for CIFAR-10. Finally, Fig. 5 shows private generated samples via DP-LDM, DP-MEPF and DP-MERF at different privacy constraints. All experiments on CIFAR-10 were conducted with a single NVIDIA V100.

Table 12: Test accuracies (higher is better) for synthetic CIFAR-10 data with varying the number of fine-tuning layers and privacy guarantees. **Top row (1-16 layers):** Fine-tuning all attention modules. **Second row (5-16 layers):** Keep first 4 attention modules fixed and fine-tuning from 5 to 16 attention modules. **Third row (9-16 layers):** Keep first 8 attention modules fixed and fine-tuning from 9 to 16 attention modules. **Bottom row (13-16 layers):** Keep first 12 attention modules fixed and fine-tuning from 13 to 16 attention modules.

DP-LDM	$\epsilon = 10$	$\epsilon = 5$	$\epsilon = 1$
1-16 layers	55.1 ± 0.3	51.2 ± 0.2	48.6 ± 0.8
5 - 16 layers	$\mathbf{63.4 \pm 0.3}$	$\mathbf{58.9 \pm 0.2}$	50.8 ± 0.5
9 - 16 layers	59.3 ± 0.5	54.9 ± 0.3	48.8 ± 0.1
13 - 16 layers	61.2 ± 0.3	56.6 ± 0.5	$\mathbf{52.3 \pm 0.5}$

Table 13: Test accuracies (higher is better) of ResNet9 trained on CIFAR-10 synthetic data with varying privacy guarantees. When trained on real data, test accuracy is 88.3%

	$\epsilon = 10$	$\epsilon = 5$	$\epsilon = 1$
DP-LDM	**63.4**	**58.9**	**50.8**
DP-MEPF (ϕ_1, ϕ_2)	48.9	47.9	28.9
DP-MEPF (ϕ_1)	51.0	48.5	29.4
DP-MERF	13.2	13.4	13.8

C.4 Transferring from Imagenet to CelebA distribution

Hyperparameter settings and FIDs for the best model trained on CelebA at 32×32 resolution with $\delta = 10^{-5}$ is supplied in Table 15. Additional experiments following Harder et al. [2023] with $\delta = 10^{-6}$ is in Table 16

Table 14: DP-LDM hyper-parameter setting on CIFAR-10. Batch sizes are lower when fine-tuning more attention modules because more gradients must be computed and stored in the backward pass. The listed values were found to fit within the memory constraints of the GPU.

		$\epsilon = 10$	$\epsilon = 5$	$\epsilon = 1$
1-16 layers (24.4M parameters)	batch size	10	10	10
	clipping norm	10^{-3}	10^{-4}	10^{-4}
	learning rate	10^{-5}	10^{-5}	10^{-5}
	epochs	50	50	100
5-16 layers (20.8M parameters)	batch size	50	50	50
	clipping norm	10^{-3}	10^{-2}	10^{-3}
	learning rate	10^{-6}	10^{-6}	10^{-5}
	epochs	50	50	10
9-16 layers (10.2M parameters)	batch size	100	100	100
	clipping norm	10^{-5}	10^{-5}	10^{-5}
	learning rate	10^{-5}	10^{-5}	10^{-5}
	epochs	10	10	10
13-16 layers (4M parameters)	batch size	200	200	200
	clipping norm	10^{-2}	10^{-2}	10^{-2}
	learning rate	10^{-6}	10^{-6}	10^{-6}
	epochs	100	100	100

Table 15: Hyperparameters and FID for finetuning on CelebA32 with $\delta = 1 \times 10^{-5}$

	$\epsilon = \infty$	$\epsilon = 10$	$\epsilon = 1$
Delta	-	10^{-5}	10^{-5}
Clipping Norm	-	0.006	0.002
Batch Size	400	250	250
Base Learning Rate	1×10^{-6}	1×10^{-6}	1×10^{-6}
Learning Rate	4×10^{-6}	2.5×10^{-4}	2.5×10^{-4}
GPU(s)	1 NVIDIA V100	1 NVIDIA V100	1 NVIDIA V100
Time	8 hours	14 hours	14 hours
FID	11.46 ± 0.20	21.0 ± 0.60	27.4 ± 0.40

Table 16: Hyperparameters and FID for finetuning on CelebA32 with $\delta = 1 \times 10^{-6}$

	$\epsilon = \infty$	$\epsilon = 10$	$\epsilon = 1$
Delta	-	10^{-6}	10^{-6}
Clipping Norm	-	0.004	0.004
Batch Size	400	150	150
Base Learning Rate	1×10^{-6}	1×10^{-6}	1×10^{-6}
Learning Rate	4×10^{-6}	1.5×10^{-4}	1.5×10^{-4}
GPU(s)	1 NVIDIA V100	1 NVIDIA V100	1 NVIDIA V100
Time	8 hours	14 hours	14 hours
FID	11.46 ± 0.20	22.39 ± 0.42	28.77 ± 0.29

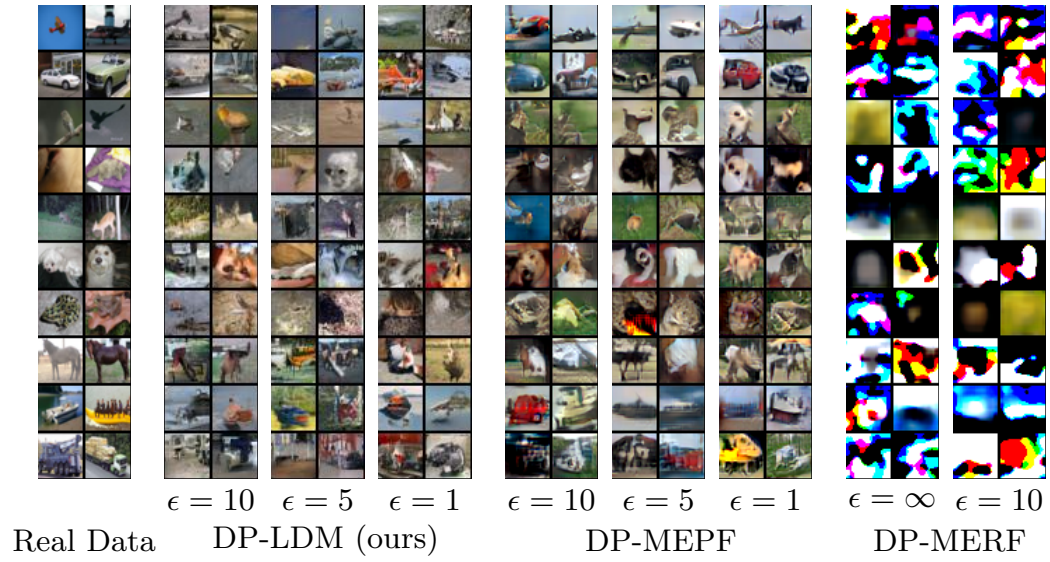

Figure 5: Labelled samples from DP-LDM, DP-MEPF (ϕ_1, ϕ_2) [Harder et al., 2023] and DP-MERF [Harder et al., 2021].