Differentially Private Dataset Condensation

Tianhang Zheng
University of Missouri-Kansas City
tzheng@umkc.edu

Baochun Li
University of Toronto
bli@ece.toronto.edu

***Abstract*—Recent work in ICML'22 established a connection between dataset condensation (DC) and differential privacy (DP), which is unfortunately problematic. To correctly connect DC and DP, we propose two differentially private dataset condensation (DPDC) algorithms—LDPDC and NDPDC. LDPDC is a linear DC algorithm that can be executed on a low-end Central Processing Unit (CPU), while NDPDC is a nonlinear DC algorithm that leverages neural networks to extract and match the latent representations between real and synthetic data. Through extensive evaluations, we demonstrate that LDPDC has comparable performance to recent DP generative methods despite its simplicity. NDPDC provides acceptable DP guarantees with a mild utility loss, compared to distribution matching (DM). Additionally, NDPDC allows a flexible trade-off between the synthetic data utility and DP budget.**

I. INTRODUCTION

Although deep learning has pushed forward the frontiers of many applications, it still needs to overcome some challenges for broader academic and commercial use [1]. One challenge is the costly process of algorithm design and practical implementation in deep learning, which typically requires inspection and evaluation by training models on massive data for many times. The growing privacy concern is another challenge. Due to the privacy concern, an increasing number of customers are reluctant to provide their data for the academia or industry to train deep learning models, and some regulations are further created to restrict access to sensitive data.

Recently, dataset condensation (DC) has emerged as a potential technique to address the two challenges with one shot [2], [3]. The main objective of dataset condensation is to condense the original dataset into a small synthetic dataset while maintaining the synthetic data utility to the greatest extent for training deep learning models. For the first challenge, both academia and industry can save computation and storage costs if using DC-generated small synthetic datasets to develop their algorithms and debug their implementations. In terms of the second challenge, since the DC-generated synthetic data may not exist in the real world, sharing DC-generated data seems less risky than sharing the original data.

Nevertheless, DC-generated data may memorize a fair amount of sensitive information during the optimization process on the original data. In other words, sharing DC-generated data still exposes the data owners to privacy risk. Moreover, this privacy risk is unknown since the prior literature on DC has not proved any rigorous connection between DC and privacy. Although a recent paper [4] claims that DC can provide certain differential privacy (DP) guarantee for free, recent work [5] demonstrates that [4] is problematic.

To correctly connect DC and DP for bounding the privacy risk of DC, we propose two differentially private dataset condensation (DPDC) algorithms—LDPDC and NDPDC. LDPDC (Algorithm 1) is a linear DC algorithm, which adds random noise to the sum of randomly sampled original data and then divides the randomized sum by the fixed expected sample size to construct the synthetic data. Based on the framework of Rényi Differential Privacy (RDP) [6], [7], we prove Theorem III.1 to bound the privacy risk of LDPDC. NDPDC (Algorithm 2) is a non-linear DC algorithm, which optimizes randomly initialized synthetic data by matching the norm-clipped representations of the synthetic data and the randomized norm-clipped representations of the original data. For NDPDC, we prove Theorem III.2 bound to its privacy risk. The potential benefits brought by DPDC algorithms include (i) reducing the cost of data storage and model training; (ii) mitigating the privacy concerns from data owners; (iii) providing a fixed DP guarantee for training multiple models on a synthetic dataset, due to the post-processing property. Besides, LDPDC is also very simple and efficient, which can be executed on a low-end CPU, allowing mobile devices to generate synthetic data.

A commonly-used method for privacy-preserving deep learning is differentially private stochastic gradient descent (DP-SGD). *Although DPDC may not achieve better model accuracy than DP-SGD, there are some use cases that can be addressed by DPDC but may not be effectively handled by DP-SGD.* One case is that the model trainers want to train and evaluate their algorithms or models on user data. If the data owners only provide DP-SGD trained models, the model trainers could not train and test their (probably private) algorithms and models on their side. If the data owners provide DPDC-generated datasets, then the model trainers can finish their tasks under a fixed DP budget. Another case is that the model trainers want to train a model on data from multiple data owners. In this case, each data owner could send a DPDC-generated dataset to the model trainer, and the model trainer could train a model on all the synthetic datasets with DP protection on each data owner's data.

We conduct extensive experiments to evaluate our DPDC algorithms on multiple datasets, including MNIST, FMNIST, CIFAR10, and CelebA. We mainly compare our DPDC algorithms with a non-private dataset condensation method, *i.e.,* distribution matching [8], and two recent differentially private

Workshop on AI Systems with Confidential Computing (AISCC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-4-5
https://dx.doi.org/10.14722/aiscc.2024.23xxx
www.ndss-symposium.org

generative methods, *i.e.,* DP-MERF and DP-Sinkhorn [9], [10]. We demonstrate that (i) LDPDC shows comparable performance to DP-MERF and DP-Sinkhorn despite its simplicity; (ii) NDPDC can provide DP protection with a mild utility loss, compared to distribution matching; (iii) NDPDC allows a flexible trade-off between privacy and utility and can use low DP budgets to achieve the overall best accuracy.

II. Background and Related Work

A. Dataset Condensation

We denote a data sample by $\boldsymbol{x}$ and its label by y. In this paper, we mainly study classification problems, where $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ refers to the model with parameters $\boldsymbol{\theta}$. $\ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ refers to the cross-entropy between the model output $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and the label y. Let $\mathcal{T}$ and $\mathcal{S}$ denote the original dataset and the synthetic dataset, respectively, then we can formulate the dataset condensation problem as

$$\arg\min_{\mathcal{S}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{T}}\ell(\boldsymbol{f}_{\boldsymbol{\theta}(\mathcal{S})}(\boldsymbol{x}), y), \tag{1}$$
$$\text{where } \boldsymbol{\theta}(\mathcal{S}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{S}}\ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), y), |\mathcal{S}| \ll |\mathcal{T}|$$

An intuitive method to solve the above objective is meta-learning [11], with an inner optimization step to update $\boldsymbol{\theta}$ and a outer optimization step to update $\mathcal{S}$. However, this intuitive method needs a lot of cost for implicitly using second-order terms in the outer optimization step. [12] considered the classification task as a ridge regression problem and derived an algorithm called kernel inducing points (KIP) to simplify the meta-learning process. Gradient matching is an alternative method [2] for dataset condensation, which minimizes a matching loss between the model gradients on the original and synthetic data, *i.e.,*

$$\min_{\mathcal{S}} \mathbb{E}_{\boldsymbol{\theta}_0\sim\mathbb{P}_{\boldsymbol{\theta}_0}}[\sum_{i=1}^{I-1}\Pi_M(\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_i), \nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}_i))]. \tag{2}$$

Π_M refers to the matching loss in [2]; $\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_i)$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}_i)$ refer to the model gradients on the synthetic and original data, respectively; $\boldsymbol{\theta_i}$ is updated on the synthetic data to obtain $\boldsymbol{\theta_{i+1}}$. To boost the performance, [3] further applied differentiable Siamese augmentation $\mathcal{A}_{\boldsymbol{w}}(\cdot)$ with parameters $\boldsymbol{w}$ to the original data samples and the synthetic data samples. Recently, [8] proposed to match the feature distributions of the original and synthetic data for dataset condensation. [8] used an empirical estimate of maximum mean discrepancy (MMD) as the matching loss, *i.e.,*

$$\mathbb{E}_{\boldsymbol{\theta}\sim P_{\boldsymbol{\theta}}}\|\frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}\boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathcal{A}_{\boldsymbol{w}}(\boldsymbol{x}_i)) - \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}\boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathcal{A}_{\boldsymbol{w}}(\boldsymbol{x}_i))\|_2^2, \tag{3}$$

where $\boldsymbol{\Phi}_{\boldsymbol{\theta}}(\cdot)$ is the feature extractor, and $P_{\boldsymbol{\theta}}$ is a parameter distribution. With the help of differentiable data augmentation [3], the distribution matching method (DM) [8] achieves the state-of-the-art performance on dataset condensation.

B. Differential Privacy

Differential Privacy (DP) [13] is the most widely-used mathematical definition of privacy, so we first introduce the definition of DP in the following.

Definition II.1 (Differential Privacy (DP))**.** For two adjacent datasets D and D', and every possible output set $\mathcal{O}$, if a randomized mechanism $\mathcal{M}$ satisfies $\mathbb{P}[\mathcal{M}(D) \in \mathcal{O}] \leq e^{\epsilon}\mathbb{P}[\mathcal{M}(D') \in \mathcal{O}] + \delta$, then $\mathcal{M}$ obeys (ϵ, δ)-DP.

Based on DP, [14] developed DP-SGD with a moments accountant for learning differentially private models.

We also introduce the concept of Rényi Differential Privacy (RDP), as RDP gives us a unified view of pure DP and (ϵ, δ)-DP, graceful composition bounds, and tighter bounds for the (sub)sampled Gaussian mechanism [7], [15]. Due to the benefits of RDP, Meta's Opacus library [16] also relies on [7] for DP analysis. We begin the brief introduction of RDP with two basic definitions:

Definition II.2 (Rényi Divergence [17])**.** Let P and Q be two distributions on $\mathcal{Z}$ over the same probability space, the Rènyi divergence between P and Q is

$$\mathcal{D}_\alpha(P\|Q) \triangleq \frac{1}{\alpha-1}\ln\int_{\mathcal{Z}} q(\boldsymbol{z})(\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})})^\alpha d\boldsymbol{z}, \tag{4}$$

where $p(\boldsymbol{z})$ and $q(\boldsymbol{z})$ are the respective probability density functions of P and Q. Without ambiguity, $\mathcal{D}_\alpha(P\|Q)$ can also be written as $\mathcal{D}_\alpha(p(\boldsymbol{z})\|q(\boldsymbol{z}))$.

Definition II.3 (Rényi Differential Privacy (RDP) [6])**.** For two adjacent datasets D and D', if a randomized mechanism $\mathcal{M}$ satisfies $\mathcal{D}_\alpha(\mathcal{M}(D)\|\mathcal{M}(D')) \leq \epsilon$ $(\alpha > 1)$, then $\mathcal{M}$ obeys (α, ϵ)-RDP, where $\mathcal{D}_\alpha$ refers to Rényi divergence.

We can easily connect RDP and DP by Lemma II.4 and Corollary II.6.

Lemma II.4 (RDP to DP Conversion [18])**.** *If a randomized mechanism $\mathcal{M}$ guarantees (α, ϵ)-RDP $(\alpha > 1)$, then it obeys $(\epsilon + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1), \delta)$-DP.*

Here we prove that Lemma II.4 is tighter than [6]'s conversion law. Therefore, we employ Lemma II.4 for conversion.

Proof: [6]'s conversion law is:

Lemma II.5 (RDP to DP Conversion [6])**.** *If a randomized mechanism $\mathcal{M}$ guarantees (α, ϵ)-RDP $(\alpha > 1)$, then it also obeys $(\epsilon + \log(1/\delta)/(\alpha-1), \delta)$-DP.*

Since $(\alpha-1)/\alpha < 1$, $\log((\alpha-1)/\alpha) < 0$. Since $\alpha > 1$, $\log\alpha > 0$, and thus $-(\log\alpha)/(\alpha-1) < 0$. Combining $\log((\alpha-1)/\alpha) < 0$ and $-(\log\alpha)/(\alpha-1) < 0$, we have

$$\log((\alpha-1)/\alpha) - (\log\alpha)/(\alpha-1) < 0 \quad \text{when } \alpha > 1. \tag{5}$$

We add $\epsilon + \log(1/\delta)/(\alpha-1)$ to both sides of the above inequality and obtain

$$\begin{aligned}\epsilon + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1) <\\ \epsilon + \log(1/\delta)/(\alpha-1) \quad \text{when } \alpha > 1.\end{aligned} \tag{6}$$

Therefore, Lemma II.4 is a tighter conversion law compared to Lemma II.5. ■

Corollary II.6. *According to Lemma II.4, if a mechanism $\mathcal{M}$ obeys $(\alpha, \epsilon(\alpha))$-RDP for $\alpha > 1$, then $\mathcal{M}$ obeys $(\min_{\alpha>1}(\epsilon(\alpha) + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1)), \delta)$-DP.*

One main advantage of RDP is that RDP allows a graceful composition of the privacy budgets spent by multiple randomized mechanisms, as illustrated in Lemma II.7.

Lemma II.7 (RDP Composition [6])**.** *If $\mathcal{M}_1$ is (α, ϵ_1)-RDP, $\mathcal{M}_2$ is (α, ϵ_2)-RDP, their composition obeys $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

Furthermore, RDP allows a graceful parallel composition, as shown in Lemma II.8.

Lemma II.8 (Parallel Composition [19])**.** *If two datasets D_1 and D_2 are disjoint ($D_1 \cap D_2 = \emptyset$), $\mathcal{M}_1$ is (α, ϵ_1)-RDP, $\mathcal{M}_2$ is (α, ϵ_2)-RDP, then the combined release $(\mathcal{M}_1(D_1), \mathcal{M}_2(D_2))$ obeys $(\alpha, \max(\epsilon_1, \epsilon_2))$-RDP for $D_1 \cup D_2$.*

We discuss more related work about differentially private generative methods in the appendix.

Algorithm 1 Linear Differentially Private Dataset Condensation (LDPDC)

Require: Original Dataset $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 ... \cup \mathcal{T}_C$; the number of classes C; number of data samples per class N_c; number of synthetic samples per class M; group size L.

- **for** each class c **do**
  - **for** $j = 1$ to M **do**
    - Take a randomly sampled subset $D_c = \{\boldsymbol{x}_k^c, c\}_{k=1}^{L_j^c}$ from $\mathcal{T}_c$ with sampling probability L/N_c (by Poisson Sampling, similar to [14], [16]) .
    - $\boldsymbol{s}_j^c = \frac{1}{L}(\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d) + \sum_{k=1}^{L_j^c} \boldsymbol{x}_k^c)$
  - **end for**
- **end for**
- Output the synthetic dataset $\mathcal{S} = \{\{\boldsymbol{s}_j^c\}_{j=1}^M\}_{c=1}^C$

III. Differentally Private Dataset Condensation (DPDC)

Dong et al. [4] tried to connect differential privacy (DP) and dataset condensation (DC), but the connection is unfortunately problematic [5]. To correctly connect DC and DP, we propose two differentially private dataset condensation (DPDC) algorithms—a linear algorithm (LDPDC) and a nonlinear algorithm (NDPDC).

A. DPDC Algorithms

a) Linear DPDC (LDPDC): We illustrate LDPDC in Algorithm 1. For each class c, we construct M synthetic data samples $\{\boldsymbol{s}_j^c\}_{j=1}^M$. For each synthetic sample $\boldsymbol{s}_j^c$, we randomly sample a subset $\{\boldsymbol{x}_k^c\}_{k=1}^{L_j^c}$ from $\mathcal{T}_c$, which is the set of all the samples with label c in $\mathcal{T}$, by Poisson Sampling with probability L/N_c. L is the group size [14], and $N_c = |\mathcal{T}_c|$. L_j^c follows a Poisson distribution with expectation L. Similar to the $q = L/N$ in [14] and the Opacus library, the sampling probability $q_c = L/N_c$ is fixed for each class in the execution of the algorithms—For the adjacent datasets of $\mathcal{T}_c$, we still consider q_c as the sampling probability, then we could exploit the prior theoretical results on subsampling for DP analysis (similar to Opacus). With $\{\boldsymbol{x}_k^c\}_{k=1}^{L_j^c}$, we compute $\boldsymbol{s}_j^c$ by $\boldsymbol{s}_j^c = \frac{1}{L}(\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d) + \sum_{k=1}^{L_j^c} \boldsymbol{x}_k^c)$ where $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d)$ refers to Gaussian random noise with standard deviation σ.

Here we assume that the real data sample $\boldsymbol{x}_k^c$ is bounded, *i.e.,* $\boldsymbol{x}_k^c \in [-b, b]^d$. For image data, after normalization, we have $b = 1$. Given this assumption, we have $\|\boldsymbol{x}_k^c\| \le b\sqrt{d}$, which bounds the sensitivity of $\sum_{k=1}^{L_j^c} \boldsymbol{x}_k^c$.

As previously noted, the strength of Algorithm 1 is not in its utility, but in its efficiency. LDPDC can be set up and executed on a low-end CPU, requiring a little computational cost to produce private synthetic data. Therefore, LDPDC is highly compatible and user-friendly for mobile devices.

Algorithm 2 Nonlinear Differentially Private Dataset Condensation (NDPDC)

Require: Original Dataset $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 ... \cup \mathcal{T}_C$; the number of classes C; the number of data samples per class N_c; the number of synthetic samples per class M; feature extractors $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$ (not pretrained); parameter distribution $P_{\boldsymbol{\theta}}$; group size L; number of iterations I.

- Initialize $\mathcal{S} = \{\{\boldsymbol{s}_j^c\}_{j=1}^M\}_{c=1}^C$ with random noise from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$
- **for** each iteration (total number of iterations is I) **do**
  - Randomly sample $\boldsymbol{\theta}$ from $P_{\boldsymbol{\theta}}$ and initialize the loss as $\ell = 0$
  - **for** each class c **do**
    - Sample the augmentation parameters $\boldsymbol{w}_c$.
    - Take a randomly sampled subset D_c from $\mathcal{T}_c$ with sampling probability L/N_c (by Poisson Sampling, similar to [14], [16]).
    - Compute Representations: $\boldsymbol{r}(\boldsymbol{x}_i^c) = \boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathcal{A}_{\boldsymbol{w}_c}(\boldsymbol{x}_i^c))$ for the subset $D_c = \{\boldsymbol{x}_i^c, c\}_{i=1}^{|D_c|}$; $\boldsymbol{r}(\boldsymbol{s}_j^c) = \boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathcal{A}_{\boldsymbol{w}_c}(\boldsymbol{s}_j^c))$ for $S_c = \{\boldsymbol{s}_j^c\}_{j=1}^M$.
    - Norm Clipping: $\tilde{\boldsymbol{r}}(\boldsymbol{s}_j^c) = \min(1, \frac{G}{\|\boldsymbol{r}(\boldsymbol{s}_j^c)\|_2})\boldsymbol{r}(\boldsymbol{s}_j^c)$; $\tilde{\boldsymbol{r}}(\boldsymbol{x}_i^c) = \min(1, \frac{G}{\|\boldsymbol{r}(\boldsymbol{x}_i^c)\|_2})\boldsymbol{r}(\boldsymbol{x}_i^c)$.
    - Compute Loss: $\ell = \ell + \| \frac{L}{M} \sum_{j=1}^M \tilde{\boldsymbol{r}}(\boldsymbol{s}_j^c) - (\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}) + \sum_{i=1}^{|D_c|} \tilde{\boldsymbol{r}}(\boldsymbol{x}_i^c)) \|_2^2$.
  - **end for**
  - $\mathcal{S} = \mathcal{S} - \eta \nabla_{\mathcal{S}} \ell \quad (\boldsymbol{s}_j^c = \boldsymbol{s}_j^c - \eta \nabla_{\boldsymbol{s}_j^c} \ell \quad \forall \boldsymbol{s}_j^c \in \mathcal{S})$.
- **end for**
- Output the synthetic dataset $\mathcal{S} = \{\{\boldsymbol{s}_j^c\}_{j=1}^M\}_{c=1}^C$

---

b) Nonlinear DPDC (NDPDC): We illustrate NDPDC in Algorithm 2, which is designed upon the idea of matching the representations of original and synthetic data. We follow [8] to use differentiable augmentation function $\mathcal{A}_{\boldsymbol{w}_c}(\cdot)$* to boost the performance ($\boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathcal{A}_{\boldsymbol{w}_c}(\cdot))$ is similar to a composite function). In each iteration of Algorithm 2, we first sample random parameters $\boldsymbol{\theta}$ for the feature extractor $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$ and initialize the loss as 0. After that, for each class c, we sample the augmentation parameters $\boldsymbol{w}_c$ and randomly sample a subset from $\mathcal{T}_c$ by Poisson sampling with sampling probability L/N_c. We then compute the representations of the original data and synthetic data and clip the representations with a pre-defined threshold G.

We remark that it is essential to clip both the representations of original and synthetic data. We clip the representations

*The transformations for augmentation include color jittering, cropping, cutout, flipping, scaling, rotation. We refer the interested readers to [3] for more details.

of the original data for the purpose of bounding the ℓ_2 sensitivity. We also clip the representations of the synthetic data in order to match the representations on a similar scale. Since G is pre-defined as the constant 1 (not computed from original data), the operation of clipping the synthetic data representations (*i.e.,* $\tilde{\boldsymbol{r}}(\boldsymbol{s}_j^c) = \min(1, \frac{G}{\|\boldsymbol{r}(\boldsymbol{s}_j^c)\|_2})\boldsymbol{r}(\boldsymbol{s}_j^c)$ in Algorithm 2) does not leak private information regarding the original data. After clipping the representations, we add Gaussian noise to the sum of the clipped original data representations.

We use the squared ℓ_2 distance between the randomized sum of the clipped original data representations (*i.e.,* $\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}) + \sum_{i=1}^{|D_c|}\tilde{\boldsymbol{r}}(\boldsymbol{x}_i^c)$) and the sum of the clipped synthetic data representations multiplied by a factor L/M (*i.e.,* $\frac{L}{M}\sum_{j=1}^{M}\tilde{\boldsymbol{r}}(\boldsymbol{s}_j^c)$) as the loss. We use the factor L/M because $\sum_{i=1}^{|D_c|}\tilde{\boldsymbol{r}}(\boldsymbol{x}_i^c)$ sums up $|D_c|$ ($\mathbb{E}(|D_c|) = L$) representations, while $\sum_{j=1}^{M}\tilde{\boldsymbol{r}}(\boldsymbol{s}_j^c)$ sums up M representations. At the end of each iteration, we update the synthetic data $\mathcal{S}$ with the gradient of the loss ℓ with respect to $\mathcal{S}$, similar to Algorithm 1 in [8]. *In practical implementation*, following [2], [3], [8], we implement $\mathcal{S}$ as a tensor variable with size $[N$, data_shape$]$ (e.g., $[N, 3, 32, 32]$ on CIFAR10), where N is the size of the synthetic dataset. *The sample diversity is ensured through the random initialization of the synthetic data samples, leading to distinct gradients for updating those samples.*

Here a natural question to ask is—Why not combine distribution matching and DP-SGD for differentially private data condensation? For common deep learning tasks, we could compute sample-wise loss so that DP-SGD can clip the sample-wise loss gradients to bound the sensitivity. However, distribution matching uses the squared ℓ_2 distance between the averaged original data representations and the averaged synthetic data representations as loss, so we could not directly compute sample-wise loss.

B. Theoretical Results

Theorem III.1 and Theorem III.2 bound the privacy risk of Algorithm 1 and Algorithm 2, respectively.

Theorem III.1. *Suppose the original data $\boldsymbol{x}$ satisfies $\boldsymbol{x} \in [-b, b]^d$, and let $\Omega_{q,\tilde{\sigma}_1}(\alpha) \triangleq \mathcal{D}_\alpha((1-q)\mathcal{N}(0, \tilde{\sigma}_1^2) + q\mathcal{N}(1, \tilde{\sigma}_1^2)\|\mathcal{N}(0, \tilde{\sigma}_1^2))$ with $\tilde{\sigma}_1 = \sigma/(b\sqrt{d})$ and $q = \max_c(L/N_c)$, then LDPDC obeys $(\alpha, M\Omega_{q,\tilde{\sigma}_1}(\alpha))$-RDP and $(\min_{\alpha>1}(M\Omega_{q,\tilde{\sigma}_1}(\alpha) + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1)), \delta)$-DP.*

Theorem III.2. *Let $\Omega_{q,\tilde{\sigma}_2}(\alpha) \triangleq \mathcal{D}_\alpha((1-q)\mathcal{N}(0, \tilde{\sigma}_2^2) + q\mathcal{N}(1, \tilde{\sigma}_2^2)\|\mathcal{N}(0, \tilde{\sigma}_2^2))$ with $\tilde{\sigma}_2 = \sigma/G$ and $q = \max_c(L/N_c)$, then NDPDC obeys $(\alpha, I\Omega_{q,\tilde{\sigma}_2}(\alpha))$-RDP and $(\min_{\alpha>1}(I\Omega_{q,\tilde{\sigma}_2}(\alpha) + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1)), \delta)$-DP, where I is the number of iterations.*

In the following, we provide the proof sketch of the above two theorems.

a) Proof Sketch: To analyze the RDP bounds of LDPDC and NDPDC, we formulate the steps that use original data samples in Algorithm 1 and 2 as Sampled Gaussian Mechanism (SGM), which is defined as

$$SGM_{q,\sigma}(D) = \boldsymbol{f}(\tilde{D}) + \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}^n).$$

$\tilde{D}$ is a subset sampled from the original dataset by Poisson sampling with sampling probability q. We then could use the following lemma to provide RDP bounds.

Lemma III.3 (RDP of SGM [7])**.** *If for any two adjacent subsets D_1 and D_2 sampled from D, $\|\boldsymbol{f}(D_1) - \boldsymbol{f}(D_2)\| \leq 1$, then $SGM_{q,\sigma}(D)$ obeys (α, ϵ)-RDP, where $\epsilon = \mathcal{D}_\alpha((1-q)\mathcal{N}(0, \sigma^2) + q\mathcal{N}(1, \sigma^2)\|\mathcal{N}(0, \sigma^2))$.*

For simplicity, in the following, we denote $\mathcal{D}_\alpha((1-q)\mathcal{N}(0, \sigma^2) + q\mathcal{N}(1, \sigma^2)\|\mathcal{N}(0, \sigma^2))$ by $\Omega_{q,\sigma}(\alpha)$. The step of using the real data samples in Algorithm 1 is

$$\boldsymbol{s}_j^c = \frac{1}{L}(\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}_d) + \sum_{k=1}^{L_j^c}\boldsymbol{x}_k^c) \tag{7}$$

If we define $\boldsymbol{g}(D) = \sum_{i=1}^{|D|}\boldsymbol{x}_i$, Eq. 7 can be formulated as an SGM, *i.e.,*

$$\boldsymbol{s}_i^c = \frac{b\sqrt{d}}{L}(\frac{1}{b\sqrt{d}}\boldsymbol{g}(D_c) + \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}_1^2\boldsymbol{I})), \text{ where } \tilde{\sigma}_1 = \sigma/(b\sqrt{d})$$

Since the sensitivity of $\frac{1}{b\sqrt{d}}\boldsymbol{g}(D_c)$ is 1, Eq. 7 guarantees $(\alpha, \Omega_{q_c,\tilde{\sigma}_1}(\alpha))$-RDP for $\mathcal{T}_c$. For $\mathcal{T}_c$, we execute Eq. 7 for M times to obtain M synthetic samples. Therefore, Algorithm 1 guarantees $(\alpha, M\Omega_{q_c,\tilde{\sigma}_1}(\alpha))$-RDP for $\mathcal{T}_c$.

Remark III.4. In practice, we release the label of $\boldsymbol{s}_i^c$, which does not affect the RDP bound for $\mathcal{T}_c$. This is because the label of all the samples in $\mathcal{T}_c$ is c, thus the additional label information does not help the adversary distinguish between $\mathcal{T}_c$ and its adjacent dataset $\mathcal{T}_c \cup \{(\boldsymbol{x}, c)\}$ for any $\boldsymbol{x}$.

The part of the optimization step that uses the real data samples in Algorithm 2 is

$$\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}) + \sum_{i=1}^{|D_c|}\tilde{\boldsymbol{r}}(\boldsymbol{x}_i^c), \tag{8}$$

where $\tilde{\boldsymbol{r}}(\boldsymbol{x}_i^c) = \min(1, \frac{G}{\|\boldsymbol{r}(\boldsymbol{x}_i^c)\|_2})\boldsymbol{r}(\boldsymbol{x}_i^c)$. If we define $\boldsymbol{g}(D) = \sum_{i=1}^{|D|}\tilde{\boldsymbol{r}}(\boldsymbol{x}_i) = \sum_{i=1}^{|D|}\min(1, \frac{G}{\|\boldsymbol{r}(\boldsymbol{x}_i)\|_2})\boldsymbol{r}(\boldsymbol{x}_i)$, then Eq. 8 can be formulated as a standard SGM, *i.e.,*

$$G(\mathcal{N}(\boldsymbol{0}, \tilde{\sigma}_2^2\boldsymbol{I}) + \frac{1}{G}\boldsymbol{g}(D_c)), \text{ where } \tilde{\sigma}_2 = \sigma/G. \tag{9}$$

Since $\boldsymbol{g}(D) = \sum_{i=1}^{|D|}\min(1, \frac{G}{\|\boldsymbol{r}(\boldsymbol{x}_i)\|_2})\boldsymbol{r}(\boldsymbol{x}_i)$, the sensitivity of $\frac{1}{G}\boldsymbol{g}(D_c)$ is also 1. Therefore, Eq. 8 guarantees $(\alpha, \Omega_{q_c,\tilde{\sigma}_2}(\alpha))$-RDP for $\mathcal{T}_c$. Since Algorithm 2 executes Eq. 8 for I iterations, it guarantees $(\alpha, I\Omega_{q_c,\tilde{\sigma}_2}(\alpha))$-RDP for $\mathcal{T}_c$.

Since $M\Omega_{q_c,\tilde{\sigma}_1}(\alpha) \leq M\max_c(\Omega_{q_c,\tilde{\sigma}_1}(\alpha))$ and $I\Omega_{q_c,\tilde{\sigma}_2}(\alpha) \leq I\max_c(\Omega_{q_c,\tilde{\sigma}_2}(\alpha))$, Algorithm 1 also guarantees $(\alpha, \max_c(M\Omega_{q_c,\tilde{\sigma}_1}(\alpha)))$-RDP, and Algorithm 2 also guarantees $(\alpha, \max_c(I\Omega_{q_c,\tilde{\sigma}_2}(\alpha)))$-RDP.

Eventually, we need to use the following corollary to conclude the proof.

Corollary III.5. *Let $\Omega_{q_c,\sigma}(\alpha) \triangleq \mathcal{D}_\alpha((1-q_c)\mathcal{N}(0, \sigma^2) + q_c\mathcal{N}(1, \sigma^2)\|\mathcal{N}(0, \sigma^2))$, where $c = 1, 2, ..., C$. We have $\max_c\Omega_{q_c,\sigma}(\alpha) = \Omega_{\max_c(q_c),\sigma}(\alpha)$.*

Based on Corollary III.5, which is proved in the appendix, we further have for $\mathcal{T}_c$, Algorithm 1 guarantees $(\alpha, M\Omega_{q,\tilde{\sigma}_1}(\alpha))$-RDP, and Algorithm 2 guarantees

$(\alpha, I\Omega_{q,\tilde{\sigma}_2}(\alpha))$-RDP, where $q = \max_c(q_c) = \max_c(L/N_c)$. Since $\mathcal{T}_1, \mathcal{T}_2, ... \mathcal{T}_C$ are disjoint, according to Lemma II.8, Algorithm 1 guarantees $(\alpha, M\Omega_{q,\tilde{\sigma}_1}(\alpha))$-RDP, and Algorithm 2 also guarantees $(\alpha, I\Omega_{q,\tilde{\sigma}_2}(\alpha))$-RDP, for $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 ... \cup \mathcal{T}_C$.

b) Additional Explanation About Releasing Labels: Remark III.4 indicates that we could release labels. Beyond Remark III.4, here we provide more explanation about the correctness of our theorems when considering labels, from the perspective of the add/remove neighboring differential privacy definition. Specifically, supposing that the adversary wants to distinguish between two adjacent datasets $\mathcal{T}$ and $\mathcal{T} \cup \{(\boldsymbol{x}, c)\}$, distinguishing between them is equivalent to distinguishing between $\mathcal{T}_c$ and $\mathcal{T}_c \cup \{(\boldsymbol{x}, c)\}$. In another word, the question of whether $(\boldsymbol{x}, c)$ is in the original dataset for dataset condensation is equivalent to the question of whether $(\boldsymbol{x}, c)$ is in the subset (with label c) of the original dataset. This is because $(\boldsymbol{x}, c)$ cannot be in the remaining part (with other labels) of the original dataset.

To distinguish between $\mathcal{T}_c$ and $\mathcal{T}_c \cup \{(\boldsymbol{x}, c)\}$, the information leakage source that the adversary can rely on is $\{\boldsymbol{s}_c^j, c\}_{j=1}^M$ since the other synthetic data is not related to $\mathcal{T}_c$. For $\mathcal{T}_c$, releasing $\{\boldsymbol{s}_c^j, c\}_{j=1}^M$ guarantees the same DP bound as releasing $\{\boldsymbol{s}_c^j\}_{j=1}^M$ since c is a constant for the samples in $\mathcal{T}_c$. To be more specific, for any dataset, we can assign a constant for all the samples, and the DP bound on the dataset should be unchanged. Therefore, for LDPDC and NDPDC, releasing $\{\boldsymbol{s}_c^j, c\}_{j=1}^M$ respectively guarantees $(\alpha, M\Omega_{q_c,\tilde{\sigma}_1}(\alpha))$-RDP and $(\alpha, I\Omega_{q_c,\tilde{\sigma}_2}(\alpha))$-RDP when the adversary distinguishes between $\mathcal{T}$ and $\mathcal{T} \cup \{(x, c)\}$, which also respectively guarantees $(\alpha, M\Omega_{q,\tilde{\sigma}_1}(\alpha))$-RDP and $(\alpha, I\Omega_{q,\tilde{\sigma}_2}(\alpha))$-RDP since $q_c = L/N_c \leq q = \max_c(L/N_c)$.

c) Additional Technical Details: In the experiments, we use $\{\{\boldsymbol{s}_c^j, c\}_{j=1}^M\}_{c=1}^C$ for training the classification models. According to Remark III.4 and the above explanation, releasing the labels does not affect the RDP bound. Additionally, we follow Opacus to exploit [7]'s method for computing $\Omega_{q,\sigma}(\alpha)$ in practice.

IV. Comparison with DPMix and DP-MERF

a) LDPDC and DPMix: Similar to LDPDC, DPMix [20] is a linear algorithm for differentially private data generation. However, LDPDC does not need to randomize the labels with the help of Lemma II.8 and Remark III.4, but DPMix needs to randomize the labels. We note that noisy labels may mislead model training and thus hurt the performance. Moreover, LDPDC takes advantage of privacy amplification by Poisson sampling, where the sampling probability is fixed, whereas the number of samples in the randomly sampled subset, *i.e.,* L_j^c, is not fixed. In contrast, DPMix claims to take use of privacy amplification by subsampling without replacement, which computes the mean over the subset with a fixed number of samples and then adds noise. According to [21], DP definition works more naturally with Poisson sampling, and Poisson sampling usually provides a better bound. Thus, DPMix needs a little bit more noise on the samples than LDPDC to achieve the same DP budget. We also note that, Poisson sampling is the standard sampling method used in the state-of-the-art Pytorch library for differentially private deep learning [16].

In addition, we have reproduced DPMix and observed that LDPDC has better performance than DPMix with the settings for dataset condensation. Specifically, for DPMix, we set $\sigma_X = \sigma_Y = 1$, $L = 50$, and $M = 50$, the result that we reproduce for DPMix is only 10.22% ± 1.23% on CIFAR10. We conjecture that this may be because (i) The operations of adding noise to the labels in DPMix cause more negative effects on model performance, compared to LDPDC. (ii) DPMix may be only able to use a large number of synthetic samples to achieve the results reported in [20]. (iii) There are some missing details in the published version of [20] that may affect the effectiveness of DPMix. If we instead refer to the results reported in [20], we also observe that LDPDC can use lower DP budgets to achieve comparable performance to DPMix. Since DPMix does not have comparable performance to recent DP generation methods and LDPDC, it is not included in the baselines in Section V.

b) NDPDC and DP-MERF: Similar to NDPDC, DP-MERF [10] proposes to match the features of the real and synthetic data. However, DP-MERF uses random Fourier features, while NDPDC uses non-linear convolutional neural networks (CNN) to extract the features. The features extracted by CNN are better representations than random Fourier features in computer vision tasks, so it is not surprising that NDPDC outperforms DP-MERF. Beyond that, DP-MERF involves label information such as one-hot label representations and randomized label counts into the embeddings before applying noise, so it seems unclear if any label information is corrupted in the process of training the generator for DP-MERF. In contrast, it is clear that NDPDC does not corrupt label information, according to Algorithm 2. Moreover, DP-MERF trains a generator to generate synthetic data, while NDPDC directly optimizes the small synthetic dataset and thus can obtain synthetic data with higher quality.

V. Experiments

A. Experimental Setup

We follow [3], [4], [8] to conduct experiments on four widely-used datasets: MNIST [22], Fashion-MNIST [23], CIFAR10 [24], and CelebA (gender classification) [25]. In the following, we introduce the baselines, DPDC settings, and the method for evaluating the synthetic data utility.

a) Baselines: We compare DPDC with the state-of-the-art dataset condensation method, *i.e.,* distribution matching (DM) [8], and two recent DP generative methods, *i.e.,* DP-Sinkhorn [9] and DP-MERF [10]. For DP-Sinkhorn, we use [9]'s code† to run the experiments. We set m to 1 and the target ϵ to 10. For DP-MERF, we use [10]'s code ‡ and follow [10] to set σ to 5.

b) DPDC Settings: For LDPDC, we set $\sigma = \sqrt{d}$, $M = 50$, $L = 50$ by default. Given that $b = 1$, $\tilde{\sigma}_1 = \sigma/(b\sqrt{d}) = 1$. For NDPDC, the default settings are $\sigma = 1$, $G = 1$, $M = 50$, $L = 50$, $I = 10000$, and $\eta = 1$. We set $G = 1$, so $\tilde{\sigma}_2 = \sigma/G = 1$. We follow [8] to use three-layer convolutional neural networks as the feature extractors (also called ConvNet-based feature extractors) for NDPDC. Batch normalization (BN) [26] is not DP friendly since a sample's

†https://github.com/nv-tlabs/DP-Sinkhorn_code

‡https://github.com/ParkLabML/DP-MERF/tree/master/code_balanced

Dataset →	MNIST		FMNIST	
Method ↓	Test Acc	DP Budget	Test Acc	DP Budget
DM with Rand Init	$98.38\% \pm 0.05\%$	No	$86.90\% \pm 0.44\%$	No
DP Sinkhorn	$86.92\% \pm 0.93\%$	$(10, 10^{-5})$-DP	$65.60\% \pm 1.06\%$	$(10, 10^{-5})$-DP
DP-MERF	$84.81\% \pm 2.04\%$	$(1, 10^{-5})$-DP	$63.05\% \pm 2.05\%$	$(1, 10^{-5})$-DP
Linear DPDC (LDPDC)	$85.79\% \pm 0.81\%$	$(1.10, 10^{-5})$-DP	$63.95\% \pm 0.42\%$	$(1.06, 10^{-5})$-DP
Nonlinear DPDC (NDPDC)	$97.35\% \pm 0.13\%$	$(6.12, 10^{-5})$-DP	$82.72\% \pm 0.35\%$	$(5.45, 10^{-5})$-DP
Dataset →	CIFAR10		CelebA	
Method ↓	Test Acc	DP Budget	Test Acc	DP Budget
DM with Rand Init	$59.69\% \pm 0.44\%$	No	$84.13\% \pm 0.42\%$	No
DP Sinkhorn	$15.09\% \pm 0.33\%$	$(10, 10^{-5})$-DP	$71.72\% \pm 1.13\%$	$(10, 10^{-5})$-DP
DP-MERF	$17.10\% \pm 0.78\%$	$(1, 10^{-5})$-DP	$69.26\% \pm 0.90\%$	$(1, 10^{-5})$-DP
Linear DPDC (LDPDC)	$25.81\% \pm 0.52\%$	$(1.14, 10^{-5})$-DP	$68.72\% \pm 2.26\%$	$(0.61, 10^{-5})$-DP
Nonlinear DPDC (NDPDC)	$52.68\% \pm 0.40\%$	$(6.72, 10^{-5})$-DP	$80.66\% \pm 0.63\%$	$(0.71, 10^{-5})$-DP

TABLE I: Here we use the default settings for all the methods. We employ 50 synthetic samples per class to train ConvNet models and report the testing accuracy here. We follow [8] to apply the augmentations in [3] when training ConvNet models. Similar to DP-Sinkhorn, we can also fix a target DP budget and compute the corresponding σ (or I) to run LDPDC and NDPDC. **In Table II, we compare all the methods under the $\epsilon = 1$ DP budget.**

	MNIST	FMNIST	CIFAR10	CelebA
Linear DPDC (LDPDC)	$85.80\% \pm 0.39\%$	$63.64\% \pm 0.76\%$	$25.46\% \pm 0.83\%$	$69.64\% \pm 1.01\%$
Nonlinear DPDC (NDPDC)	$95.32\% \pm 0.29\%$	$78.79\% \pm 0.37\%$	$42.40\% \pm 0.86\%$	$81.47\% \pm 0.80\%$
DP Sinkhorn	$55.43\% \pm 1.54\%$	$43.22\% \pm 1.40\%$	$12.62\% \pm 0.27\%$	$64.02\% \pm 0.48\%$
DP-MERF	$84.81\% \pm 2.04\%$	$63.05\% \pm 2.05\%$	$17.10\% \pm 0.78\%$	$69.26\% \pm 0.90\%$

TABLE II: **We set $\epsilon = 1, \delta = 10^{-5}$ and compare LDPDC, NDPDC, DP Sinkhorn, DP-MERF.**

normalized value depends on other samples [16]. Therefore, we do not use BN in the extractors. Since the data statistics like channel-wise means and channel-wise standard deviation may leak private information, we follow [9] to use a fixed value 0.5 for normalizing the images, which does not make obvious difference in the performance of DPDC and the baselines. After normalization, the pixel values range from -1 to 1.

c) Performance Evaluation: We employ the evaluation method in [3], [4], [8] to compare the performance of DM, DP-Sinkhorn, DP-MERF, and our DPDC algorithms. The evaluation method is to train deep learning models on the synthetic data from scratch and compare their accuracy on the real testing data. Higher testing accuracy indicates better synthetic data utility for training deep learning models (better performance). We also follow [3], [8] to train a variety of model architectures, including MLP [27], LeNet [28], AlexNet [29], VGG11 [30], and ResNet18 [31], on DPDC-generated synthetic data to evaluate the data utility.

B. Main Results

We report the main experimental results in Table I & II. Our LDPDC algorithm achieves comparable performance to DP-MERF and DP-Sinkhorn with low DP budgets. Our NDPDC algorithm provides acceptable DP guarantees ($\epsilon < 10$) with a mild utility loss, compared to the random-initialized non-private DM method [8]. Furthermore, NDPDC allows a flexible trade-off between synthetic data utility and DP budget—If we are not satisfied with NDPDC's DP budgets in Table I, we could increase σ to reduce the budgets. **As shown in Table II, even with low DP budgets like $\epsilon = 1$, NDPDC still outperforms LDPDC, DP-Sinkhorn, and DP-MERF.** For DP-MERF, even if we decrease σ to 0.5 ($\epsilon > 10$), the accuracy increment is only about 7% on FMNIST and less than 5% on the other datasets, as shown in Table IV. We conjecture that NDPDC-condensed synthetic data is more useful than DP-generator generated synthetic data because DP generative methods optimize the generative model parameters, while NDPDC directly optimizes the small amount of synthetic data. Moreover, NDPDC have other advantages over DP-MERF as discussed in Section IV.

We further train a variety of model architectures on the synthetic data generated by LDPDC and NDPDC and report the testing accuracy in Table III. Since LDPDC does not rely on deep networks to learn the synthetic data, it is hard to predict which network architecture can make the best use of the LDPDC-generated synthetic data. According to the results in Table III, on FMNIST, CIFAR10, and CelebA, MLP makes the best use of the simple LDPDC-generated synthetic data, while on MNIST, ResNet18 makes the best use of the synthetic data. For NDPDC, since the synthetic data is learned on ConvNet-based extractors, ConvNet makes better use of the synthetic data than the other architectures. We also evaluate DP-HP [32] using [32]'s code[§] on MNIST and FMNIST and report the results in Table V. Different from DP-Sinkhorn and DP-MERF, we do not obtain comparable results for DP-HP, so DP-HP is not included in Table I.

C. Visualization

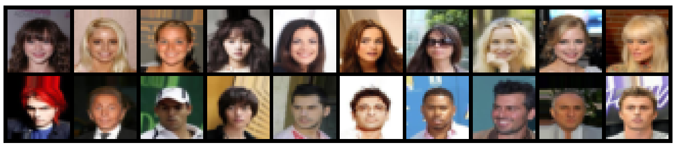
We visualize synthetic images generated by DM, DP-Sinkhorn, DP-MERF, LDPDC, and NDPDC in Fig. 1. For DPMix, interested readers can refer to Fig. 1 in [20] for the synthetic images generated by DPMix.

Compared to DM-generated images, NDPDC-generated images are more noisy due to the DP guarantees. A surprising result is that LDPDC-generated images look like noise, but the models still can learn some information from LDPDC-generated images. We conjecture that this is because LDPDC-generated images still have certain patterns. However, the

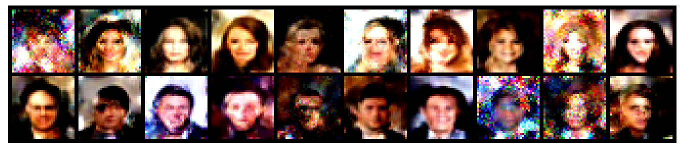
[§]https://github.com/ParkLabML/DP-HP

LDPDC	MLP	LeNet	AlexNet	VGG11	ResNet18
MNIST	$80.40\% \pm 0.33\%$	$83.32\% \pm 3.03\%$	$83.22\% \pm 0.67\%$	$85.81\% \pm 0.74\%$	$88.14\% \pm 0.73\%$
FMNIST	$68.84\% \pm 0.58\%$	$68.49\% \pm 1.36\%$	$66.45\% \pm 0.94\%$	$67.31\% \pm 1.01\%$	$67.58\% \pm 0.92\%$
CIFAR10	$30.60\% \pm 0.18\%$	$29.82\% \pm 0.81\%$	$29.65\% \pm 0.82\%$	$24.86\% \pm 0.29\%$	$23.79\% \pm 0.59\%$
CelebA	$69.48\% \pm 1.51\%$	$66.95\% \pm 0.53\%$	$67.61\% \pm 0.87\%$	$66.87\% \pm 2.23\%$	$65.88\% \pm 1.42\%$
NDPDC	**MLP**	**LeNet**	**AlexNet**	**VGG11**	**ResNet18**
MNIST	$93.15\% \pm 0.83\%$	$96.82\% \pm 0.13\%$	$97.00\% \pm 0.32\%$	$97.37 \pm 0.13\%$	$97.67\% \pm 0.14\%$
FMNIST	$79.98\% \pm 0.33\%$	$80.89\% \pm 0.40\%$	$81.94\% \pm 0.54\%$	$82.63\% \pm 0.54\%$	$81.50\% \pm 0.33\%$
CIFAR10	$36.99\% \pm 0.62\%$	$36.94\% \pm 1.24\%$	$42.60\% \pm 1.06\%$	$48.80\% \pm 0.24\%$	$44.79\% \pm 0.64\%$
CelebA	$75.95\% \pm 1.30\%$	$78.02\% \pm 1.01\%$	$79.69\% \pm 0.91\%$	$81.25\% \pm 1.31\%$	$82.32\% \pm 0.68\%$

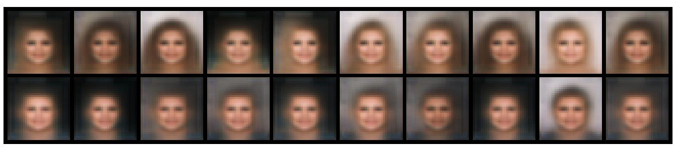
TABLE III: Performance on varied model architectures with default settings: For NDPDC, the synthetic data is learned on ConvNet-based feature extractors and evaluated on those model architectures. **The privacy budgets and the results on ConvNet are given in Table I.**



(a) Original images.

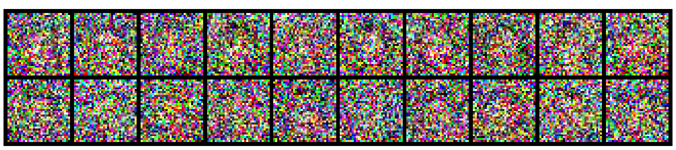


(b) DM generated synthetic images (with random initialization).



(c) DP-Sinkhorn generated synthetic images

(d) DP-MERF generated synthetic images.



(e) LDPDC generated synthetic images.

(f) NDPDC generated synthetic images.

Fig. 1: Visualizing the synthetic CelebA images. Female synthetic images are listed in the first row, and male synthetic images are listed in the second row.

Dataset	Test Acc	DP budget
MNIST	$86.70\% \pm 2.07\%$	$(11.60, 10^{-5})$-DP
FMNIST	$70.38\% \pm 0.79\%$	$(11.60, 10^{-5})$-DP
CIFAR10	$20.61\% \pm 0.87\%$	$(11.60, 10^{-5})$-DP
CelebA	$69.51\% \pm 1.69\%$	$(11.60, 10^{-5})$-DP

TABLE IV: The performance of DP-MERF with $\sigma = 0.5$. We employ 50 synthetic samples per class to train the ConvNet models for evaluation.

Dataset	Test Acc	DP budget
MNIST	$74.20\% \pm 1.66\%$	$(1, 10^{-5})$-DP
FMNIST	$28.05\% \pm 1.12\%$	$(1, 10^{-5})$-DP

TABLE V: The performance of DP-HP. We run the experiments using [32]'s code to generate synthetic data. We employ 50 synthetic samples per class to train the ConvNet models for evaluation.

patterns are hardly perceptible by human beings because of the high noise ($\sigma = \sqrt{d}$).

We remark that, although the synthetic images generated by DP-Sinkhorn on CelebA look like faces, they are not very colorful and not diverse. Therefore, when being tested on the colorful and diverse original CelebA images, the model trained on NDPDC-generated images has better accuracy than the model trained on DP-Sinkhorn generated images.

D. Case Study on Tabular Data

Previous research on dataset condensation [2], [4], [8] mainly focuses on image data. We conjecture that the reason is that most tabular datasets are already small and may not need condensation. Nevertheless, in this paper, we conduct a case study on a widely used tabular dataset, *i.e.,* adult income dataset [33], to evaluate our DPDC methods on diverse data formats. In this case study, we use a single layer linear network (14×64) as the feature extractor for NDPDC. We still set $M = 50$, so the total number of synthetic samples is 100. We set $\epsilon = 1$, which results in $\sigma = 2.09$. The testing accuracy achieved by NDPDC on the adult income dataset is $79.16\% \pm 0.09\%$. We also run DP-MERF with the setting of 100 synthetic training samples, and the best accuracy we can obtain for DP-MERF is 70.60%.

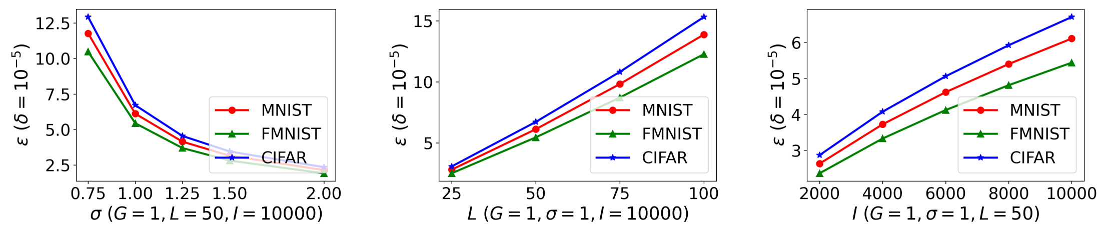


Fig. 2: The privacy budgets of NDPDC with different σ, L, and I.

Noise	MNIST	FMNIST	CIFAR10
$\sigma = 0.75$	97.51%	83.28%	54.33%
$\sigma = 1.25$	97.06%	82.23%	51.36%
$\sigma = 1.5$	96.86%	81.66%	50.12%
$\sigma = 2.0$	96.46%	80.96%	47.93%

TABLE VI: The averaged testing accuracy of ConvNets trained on the synthetic data generated by NDPDC with different σ.

Group Size	MNIST	FMNIST	CIFAR10
$L = 25$	96.41%	80.88%	48.07%
$L = 75$	97.61%	83.97%	54.81%
$L = 100$	97.90%	84.40%	56.42%

TABLE VII: The averaged testing accuracy of ConvNets trained on the synthetic data generated by NDPDC with different group size L.

Iterations	MNIST	FMNIST	CIFAR10
$I = 2000$	96.32%	80.45%	47.42%
$I = 4000$	96.84%	81.64%	50.07%
$I = 6000$	97.14%	82.18%	51.20%
$I = 8000$	97.28%	82.50%	52.19%

TABLE VIII: The averaged testing accuracy of ConvNets trained on NDPDC-generated synthetic data with different I.

Data Size	MNIST	FMNIST	CIFAR10
$M = 10$	96.47%	80.00%	45.88%
$M = 100$	97.54%	82.88%	54.24%
$M = 200$	97.50%	83.16%	54.26%

TABLE IX: The averaged testing accuracy of ConvNets achieved by NDPDC with different number of synthetic samples per class M.

E. Ablation Study on NDPDC

Although LDPDC is simple and comparable to recent DP generative methods here, we still recommend the readers with sufficient computational resources to use NDPDC because of its outstanding performance. In this subsection, we conduct ablation study for NDPDC on MNIST, FMNIST, and CIFAR10 with recommendations on how to select hyperparameters for executing NDPDC. When we study the effects of one hyperparameter, we fix the other hyperparameters as the default settings.

a) Effects of Noise Multiplier σ on Privacy and Utility: We plot the DP budgets of NDPDC with different σ in Fig. 2 and report the corresponding testing accuracy in Table VI. Fig. 2 and Table VI indicate that, as σ increases, ϵ will decrease, and the testing accuracy will also decrease. But the testing accuracy does not decrease much as σ increases. Thus, if we are unsatisfied with the DP budget, we could simply increase σ to obtain a low DP budget with a little loss of synthetic data utility. Additionally, we do not recommend to set $\sigma/G \leq 0.75$, otherwise ϵ will be larger than 10, then the DP guarantee is not very useful.

b) Effects of Group Size L on Privacy and Utility: We plot the DP budgets with different L in Fig. 2 and show the testing accuracy in Table VII. If we increase L, more original data will be sampled in each step for learning the synthetic data, and thus, both ϵ and the testing accuracy will increase. According to Fig. 2 and Table VII, if we are unsatisfied with the DP budgets, we could also decrease L to 25 to obtain better DP budgets with a minor utility loss.

c) Effects of Number of Iterations I on Privacy and Utility: We plot the DP budgets in Fig. 2 and show the testing accuracy in Table VIII. If we increase I, since NDPDC will learn on the original data for more iterations, both the DP budget and testing accuracy will increase. Since the testing accuracy does not increase much when I is large, reducing I is another good choice beyond increasing σ and reducing L to save DP budget with acceptable utility loss.

d) Effects of Synthetic Data Size M on Privacy and Utility: We show the testing accuracy with different M in Table IX: As M increases from 10 to 200, the testing accuracy also increases but will almost stop the uptrend around a certain M. This is probably because more synthetic data has the potential to capture more information, but the DP guarantee also limits the information leaked from the original data. Since the DP budget is unchanged, after the amount of synthetic data is enough for capturing all the limited information, more synthetic data could not capture more useful information. Given the experimental results, $M = 50$ is a good setting here. We do not recommend setting a larger M ($M > 50$), which will result in marginal performance gain but much more cost in the downstream applications.

VI. Conclusion

In this paper, we connect data condensation and differential privacy by proposing two differentially private dataset condensation (DPDC) algorithms—LDPDC and NDPDC. We demonstrate that LDPDC can use low DP budgets to achieve comparable performance to DP-Sinkhorn and DP-MERF. Moreover, we show that NDPDC can provide DP guarantees with a mild utility loss compared to the distribution matching method. We hope our work can inspire further research in this potential direction to alleviate the cost burden and privacy concern in deep learning.

References

[1] C. Dilmegani. (2022) 4 ways to overcome deep learning challenges in 2022. [Online]. Available: https://research.aimultiple.com/deep-learning-challenges/

[2] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching." *ICLR*, vol. 1, no. 2, p. 3, 2021.

[3] B. Zhao and H. Bilen, "Dataset condensation with differentiable siamese augmentation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 674–12 685.

[4] T. Dong, B. Zhao, and L. Lyu, "Privacy for free: How does dataset condensation help privacy?" *arXiv preprint arXiv:2206.00240*, 2022.

[5] N. Carlini, V. Feldman, and M. Nasr, "No free lunch in" privacy for free: How does dataset condensation help privacy"," *arXiv preprint arXiv:2209.14987*, 2022.

[6] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 2017, pp. 263–275.

[7] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled gaussian mechanism," *arXiv preprint arXiv:1908.10530*, 2019.

[8] B. Zhao and H. Bilen, "Dataset condensation with distribution matching," *CoRR*, vol. abs/2110.04181, 2021.

[9] T. Cao, A. Bie, A. Vahdat, S. Fidler, and K. Kreis, "Don't generate me: Training differentially private generative models with sinkhorn divergence," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 480–12 492, 2021.

[10] F. Harder, K. Adamczewski, and M. Park, "Dp-merf: Differentially private mean embeddings with randomfeatures for practical privacy-preserving data generation," in *International conference on artificial intelligence and statistics*. PMLR, 2021, pp. 1819–1827.

[11] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.

[12] T. Nguyen, Z. Chen, and J. Lee, "Dataset meta-learning from kernel ridge-regression," in *International Conference on Learning Representations*, 2020.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[14] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[15] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1226–1235.

[16] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao *et al.*, "Opacus: User-friendly differential privacy library in pytorch," *arXiv preprint arXiv:2109.12298*, 2021.

[17] A. Rényi *et al.*, "On measures of entropy and information," in *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 547-561. Berkeley, California, USA, 1961.

[18] B. Balle, G. Barthe, M. Gaboardi, J. Hsu, and T. Sato, "Hypothesis testing interpretations and renyi differential privacy," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2496–2506.

[19] K. Chaudhuri, J. Imola, and A. Machanavajjhala, "Capacity bounded differential privacy," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] K. Lee, H. Kim, K. Lee, C. Suh, and K. Ramchandran, "Synthesizing differentially private datasets using random mixing," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 542–546.

[21] Y. Zhu and Y.-X. Wang, "Poission subsampled rényi differential privacy," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7634–7642.

[22] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[23] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[27] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] M. Vinaroz, M.-A. Charusaie, F. Harder, K. Adamczewski, and M. J. Park, "Hermite polynomial features for private data generation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 22 300–22 324.

[33] B. Becker and R. Kohavi, "Adult," UCI Machine Learning Repository, 1996, DOI: https://doi.org/10.24432/C5XW20.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[36] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[37] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[38] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[39] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.

[40] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *arXiv preprint arXiv:1802.06739*, 2018.

[41] J. Jordon, J. Yoon, and M. Van Der Schaar, "Pate-gan: Generating synthetic data with differential privacy guarantees," in *International conference on learning representations*, 2018.

[42] R. Torkzadehmahani, P. Kairouz, and B. Paten, "Dp-cgan: Differentially private synthetic data and label generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[43] Y. Long, B. Wang, Z. Yang, B. Kailkhura, A. Zhang, C. Gunter, and B. Li, "G-pate: Scalable differentially private data generator via private aggregation of teacher discriminators," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2965–2977, 2021.

[44] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2016.

[45] D. Chen, T. Orekondy, and M. Fritz, "Gs-wgan: A gradient-sanitized approach for learning differentially private generators," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 673–12 684, 2020.

Appendix

Omitted Proof

We first prove Lemma A.1. Based on Lemma A.1, we can easily prove Corollary A.2 (Corollary III.5), which is used in the proof of Theorem III.1 & III.2.

Lemma A.1. *Let $u(\boldsymbol{z})$ and $\nu(\boldsymbol{z})$ be two differentiable probability density functions on a domain $\mathcal{Z}$ ($u, \nu : \mathcal{Z} \mapsto \mathbb{R}$). If $u(\boldsymbol{z}) \neq \nu(\boldsymbol{z})$ and $u(\boldsymbol{z}), \nu(\boldsymbol{z}) > 0$ on $\mathcal{Z}$, then $\mathcal{D}_\alpha((1-q)u(\boldsymbol{z}) + q\nu(\boldsymbol{z}) \| u(\boldsymbol{z}))$ is an increasing function w.r.t. q when $\alpha > 1$ and $q \in [0, 1]$.*

Lemma A.1 is easy to understand: As q increases, the weight of $u(\boldsymbol{z})$ in the mixture $(1-q)u(\boldsymbol{z}) + q\nu(\boldsymbol{z})$ decreases, thus the divergence between $(1-q)u(\boldsymbol{z}) + q\nu(\boldsymbol{z})$ and $u(\boldsymbol{z})$ should increase. *To our best knowledge, we are the first to present Lemma A.1, so we detail the proof in the following.*

proof of Lemma A.1: The Rényi divergence $\mathcal{D}_\alpha((1-q)u(\boldsymbol{z}) + q\nu(\boldsymbol{z}) \| u(\boldsymbol{z}))$ is defined as

$$\frac{1}{\alpha-1} \ln \int_{\mathcal{Z}} u(\boldsymbol{z}) (\frac{(1-q)u(\boldsymbol{z}) + q\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^\alpha d\boldsymbol{z} = \frac{1}{\alpha-1} \ln \int_{\mathcal{Z}} u(\boldsymbol{z})(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^\alpha d\boldsymbol{z} \tag{10}$$

The derivative of Eq. 10 w.r.t q is

$$\frac{\int_{\mathcal{Z}} \alpha(\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z}}{(\alpha-1)\int_{\mathcal{Z}} u(\boldsymbol{z})(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^\alpha d\boldsymbol{z}}. \tag{11}$$

To prove Lemma A.1, we need to show Eq. 11 is positive when $q \in [0, 1]$. Since $\alpha > 1$ and $u(\boldsymbol{z})(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^\alpha > 0$ (If $q \neq 0$, $1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})} > 1-q \geq 0$), we only need to prove $\int_{\mathcal{Z}}(\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z} > 0$.

We divide $\mathcal{Z}$ into $\mathcal{Z}_1$ and $\mathcal{Z}_2$, where $\mathcal{Z}_1 = \{\boldsymbol{z} \in \mathcal{Z} | \nu(\boldsymbol{z}) < u(\boldsymbol{z})\}$ and $\mathcal{Z}_2 = \{\boldsymbol{z} \in \mathcal{Z} | \nu(\boldsymbol{z}) \geq u(\boldsymbol{z})\}$. Apparently, $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are disjoint, and $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$. Thus, we can rewrite $\int_{\mathcal{Z}}(\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z}$ as

$$\int_{\mathcal{Z}_1} (\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z} + \int_{\mathcal{Z}_2} (\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z} \tag{12}$$

When $\boldsymbol{z} \in \mathcal{Z}_1$, we have (i) $\nu(\boldsymbol{z}) - u(\boldsymbol{z}) < 0$; (ii) $\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})} < 1$ ($0 < \nu(\boldsymbol{z}) < u(\boldsymbol{z})$); (iii) $0 < (1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} < 1$ ($1 = 1-q+q > 1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})} > 1-q \geq 0$). Therefore,

$$\int_{\mathcal{Z}_1} (\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z} > \int_{\mathcal{Z}_1} (\nu(\boldsymbol{z}) - u(\boldsymbol{z})) d\boldsymbol{z} \tag{13}$$

When $\boldsymbol{z} \in \mathcal{Z}_2$, we have (i) $\nu(\boldsymbol{z}) - u(\boldsymbol{z}) \geq 0$; (ii) $\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})} \geq 1$ ($0 < u(\boldsymbol{z}) \leq \nu(\boldsymbol{z})$); (iii) $(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} \geq 1$. Therefore,

$$\int_{\mathcal{Z}_2} (\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z} \geq \int_{\mathcal{Z}_2} (\nu(\boldsymbol{z}) - u(\boldsymbol{z})) d\boldsymbol{z} \tag{14}$$

As a result,

$$\int_{\mathcal{Z}} (\nu(\boldsymbol{z}) - u(\boldsymbol{z}))(1-q+q\frac{\nu(\boldsymbol{z})}{u(\boldsymbol{z})})^{\alpha-1} d\boldsymbol{z} > \int_{\mathcal{Z}_1} (\nu(\boldsymbol{z}) - u(\boldsymbol{z})) d\boldsymbol{z} + \int_{\mathcal{Z}_2} (\nu(\boldsymbol{z}) - u(\boldsymbol{z})) d\boldsymbol{z} = \int_{\mathcal{Z}} (\nu(\boldsymbol{z}) - u(\boldsymbol{z})) d\boldsymbol{z} = \int_{\mathcal{Z}} \nu(\boldsymbol{z}) d\boldsymbol{z} - \int_{\mathcal{Z}} u(\boldsymbol{z}) d\boldsymbol{z} = 0 \tag{15}$$

Thus, the derivative of Eq. 10 w.r.t q is positive. This concludes the proof of Lemma A.1. ■

Corollary A.2. *Let $\Omega_{q_c,\sigma}(\alpha) \triangleq \mathcal{D}_\alpha((1-q_c)\mathcal{N}(0,\sigma^2) + q_c\mathcal{N}(1,\sigma^2) \| \mathcal{N}(0,\sigma^2))$, where $c = 1, 2, ..., C$. We have $\max_c \Omega_{q_c,\sigma}(\alpha) = \Omega_{\max_c(q_c),\sigma}(\alpha)$.*

Proof of Corollary A.2: Let $u(z) \triangleq \mathcal{N}(0,\sigma^2)$ and $\nu(z) \triangleq \mathcal{N}(1,\sigma^2)$ ($\mathcal{Z} \triangleq R$), then based on Lemma A.1, we know that $\Omega_{q,\sigma}(\alpha)$ is an increasing function w.r.t. q. Thus, the maximum of $\Omega_{q_c,\sigma}(\alpha)$ is achieved at $\max_c(q_c)$. This concludes the proof of Corollary A.2. ■

Additional Related Work

The previous literature has proposed an array of generative methods for synthetic data generation [34]–[39]. Due to the growing privacy concern, recent research also focuses on developing differentially private generative methods [40]–[43]. [40] first combined DP-SGD and GAN to generate private synthetic data. [42] combined conditional GAN and DP-SGD to generate class-conditional private data. [41] applied PATE [44] to GAN and developed a differentially private GAN framework called PATE-GAN. PATE-GAN trains a student discriminator on the labels output by the PATE mechanism and trains the generator on the generative loss computed over the student discriminator. [43] proposed a framework called G-PATE with a private gradient aggregation mechanism to enable a better combination of PATE and GAN. GS-WGAN [45] proposed to selectively apply the randomized mechanism in DP-SGD to maximally preserve the true gradient direction and use the Wasserstein objective to improve the amount of gradient information flow during training the generative models. DP-MERF [10] proposed to train the generator by matching the mean embeddings of the real data and the generator-output synthetic data. DP-Shinkhorn [9] framed the generative learning problem as minimizing the optimal transport distance and trained the generative models using a semi-debiased Sinkhorn loss. [9] demonstrated that, using $(10, 10^{-5})$-DP budget, DP-Shinkhorn can generate synthetic data with better utility and quality than G-PATE and GS-WGAN on MNIST and FashionMNIST.