DIFFERENTIALLY PRIVATE SYNTHETIC DATA VIA APIS 3: USING SIMULATORS INSTEAD OF FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentially private (DP) synthetic data, which closely resembles the original private data while maintaining strong privacy guarantees, has become a key tool for unlocking the value of private data without compromising privacy. Recently, PRIVATE EVOLUTION (PE) has emerged as a promising method for generating DP synthetic data. Unlike other training-based approaches, PE only requires access to inference APIs from foundation models, enabling it to harness the power of state-of-the-art models. However, a suitable foundation model for a specific private data domain is not always available. In this paper, we discover that the PE framework is sufficiently general to allow inference APIs beyond foundation models. Specifically, we show that simulators—such as computer graphics-based image synthesis tools—can also serve as effective APIs within the PE framework. This insight greatly expands the applicability of PE, enabling the use of a wide variety of domain-specific simulators for DP data synthesis. We explore the potential of this approach, named SIM-PE, in the context of image synthesis. Across three diverse simulators, SIM-PE performs well, improving the downstream classification accuracy of PE by up to 3× and reducing the FID score by up to 80%. We also show that simulators and foundation models can be easily leveraged together within the PE framework to achieve further improvements.

1 INTRODUCTION

Leaking sensitive user information is a significant concern in data-driven applications. A common solution is to generate differentially private (DP) (Dwork et al., 2006) synthetic data that closely resembles the original while ensuring strict privacy guarantees. This DP synthetic data can replace the original in tasks like model fine-tuning and statistical analysis while preserving user privacy.

PRIVATE EVOLUTION (PE) (Lin et al., 2023; Xie et al., 2024) has recently emerged as a promising method for generating DP synthetic data. PE starts by probing a foundation model to generate random samples, then iteratively selects those most similar to the private data and uses the model to generate more samples that resemble them. Unlike previous state-of-the-art approaches that require fine-tuning open-source models, PE relies solely on model inference. Therefore, PE can be up to 66× faster than training-based methods (Xie et al., 2024). More importantly, this enables PE to easily harness the cutting-edge foundation models like GPT-4 (OpenAI, 2023) and Stable Diffusion (Rombach et al., 2022), achieving state-of-the-art performance across multiple image and text benchmarks (Lin et al., 2023; Xie et al., 2024; Hou et al., 2024).

However, PE relies on foundation models suited to the private data domain, which may not always be available. When the model's distribution significantly differs from the private data, PE's performance lags far behind training-based methods (Gong et al., 2025).

To address this question, we note that in the traditional synthetic data field—where private data is *not* involved—domain-specific, non-neural-network *simulators* remain widely used, especially in domains where foundation models struggle. Examples include computer graphics-based simulators for images, videos, and 3D data (e.g., Blender (Community, 2018) and Unreal (Epic Games)), physics-based simulators for robotics data (e.g., Genesis (Authors, 2024)), and network simulators for networking data (e.g., ns-2 (Issariyakul et al., 2009)). While these simulators have been successful, their applications in *DP* data synthesis remain underexplored. This is understandable, as adapting these simulators to fit private data in a DP fashion requires non-trivial, case-by-case modifications. Our key insight is that PE only requires two APIs: RANDOM_API that generates random samples and VARIATION_API that generates samples similar to the given one. These APIs do not have to come from foundation models! Thus, we ask: *Can* PE *use simulators in place of foundation*

models? If viable, this approach could greatly expand PE's capabilities and unlock the potential of a wide range of domain-specific simulators for DP data synthesis.

In this paper, we explore this potential in the context of *images*. We consider two types of simulator access: **(1) The simulator is accessible.** In this case, we define RANDOM_API as using random simulator parameters to render an image, and VARIATION_API as slightly perturbing the simulator parameter of the given image. **(2) The simulator is *not* accessible—only its generated dataset is released.** This scenario is quite common (Wood et al., 2021; Bae et al., 2023), especially when simulator assets are proprietary (Kar et al., 2019; Devaranjan et al., 2020). In this case, we define RANDOM_API as randomly selecting an image from the dataset, and VARIATION_API as randomly selecting a nearest neighbor of the given image. We demonstrate that the resulting algorithm, SIM-PE, outperforms PE with foundation models. Our key contributions are:

- **Insight:** We identify that PE is not limited to foundation models, making it the first framework capable of utilizing both state-of-the-art foundation models and simulators for DP data synthesis.
- **Algorithm:** We propose SIM-PE, an extension of PE using simulators, applicable in both scenarios where the simulator or the generated dataset is available. Additionally, we introduce the use of *both foundation models and simulators interchangeably during the data synthesis process*, allowing for the benefits of both to be leveraged through PE's easy and standardized interface.
- **Results:** We demonstrate promising results with SIM-PE. For instance, on the MNIST dataset with $\epsilon = 1$, downstream classification accuracy increases to 89.1%, compared to 27.9% with the original PE. Furthermore, combining foundation models with weak simulators results in improved performance compared to using either one alone.

2 PRELIMINARIES AND MOTIVATION

Synthetic data refers to "fake" data generated by models or software for various applications, including data augmentation, model training, and software testing (Lin, 2022). One approach involves **machine learning models**, ranging from simple statistical models like Gaussian mixtures to more advanced deep neural network-based generative models such as GANs (Goodfellow et al., 2020), diffusion models (Sohl-Dickstein et al., 2015), and auto-regressive models (OpenAI, 2023; Liu et al., 2024a). The other approach relies on **simulators**. **In this paper, we broadly define simulators as non-neural-network data synthesizers with hard-coded, interpretable logic.** For example, given network configurations, ns-2 (Issariyakul et al., 2009) can simulate a network and generate network packets. Similarly, given 3D models and lighting configurations, Blender (Community, 2018) can render images and videos of objects. These simulators are widely used and are especially useful when data distributions are too complex for machine learning models to learn.

DP synthetic data requires the synthetic data to be *close to a given private dataset*, while *having a strict privacy guarantee called Differential Privacy (DP)* (Dwork et al., 2006). Formally, a mechanism $\mathcal{M}$ is (ϵ, δ)-DP if for any two neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$ (i.e., $\mathcal{D}'$ has one extra entry compared to $\mathcal{D}$ or vice versa) and for any set S of outputs of $\mathcal{M}$, we have $\mathbb{P}\left(\mathcal{M}\left(\mathcal{D}\right) \in S\right) \leq e^{\epsilon}\mathbb{P}\left(\mathcal{M}\left(\mathcal{D}'\right) \in S\right) + \delta$. Smaller ϵ and δ imply stronger privacy guarantees. Current state-of-the-art DP image and text synthesis methods rely on machine learning models and typically require model training (Lin et al., 2020a; Beaulieu-Jones et al., 2019; Dockhorn et al., 2022; Yin et al., 2022; Yu et al., 2021; He et al., 2022; Li et al., 2021; Ghalebikesabi et al., 2023a; Yue et al., 2022; Jordon et al., 2019; Harder et al., 2023; 2021b; Vinaroz et al., 2022; Cao et al., 2021).

PRIVATE EVOLUTION (PE) (Lin et al., 2023; Xie et al., 2024) is a recent training-free framework for DP data synthesis. PE only requires inference access to the foundation models. Therefore, unlike prior training-based methods, PE can leverage the state-of-the-art models even if they are behind APIs (e.g., GPT-4) and is more computationally efficient. PE is versatile across data modalities, as long as suitable foundation models are available with two functions: (1) RANDOM_API that generates a random sample (e.g., a random bird image), and (2) VARIATION_API that generates slight modifications of the given sample (e.g., a similar bird image). PE works by first calling RANDOM_API to get an initial set of synthetic samples, and then iteratively refine this set by selecting the closest ones to the private samples (in a DP manner) and calling VARIATION_API to generate more of such samples. The full PE algorithm from Lin et al. (2023) is attached in App. A for completeness.

Motivation. While PE achieves state-of-the-art performance on several image and text benchmarks (Lin et al., 2023; Xie et al., 2024), its performance significantly drops when there is a large distribution shift between the private data and the foundation model's pre-trained data (Gong et al.,

2025). Since relevant foundation models may not always be available for every domain, this limitation hinders PE's applicability in real-world scenarios. Extending PE to leverage simulators could significantly expand its potential applications.

More broadly, as discussed above, simulators cannot be substituted by foundation models in (non-DP) data synthesis across many domains. Unfortunately, current state-of-the-art DP synthetic data methods are deeply reliant on machine learning models (e.g., requiring model training) and cannot be applied to simulators. By extending PE to work with simulators, we aim to unlock the potential of simulators in DP data synthesis.

3 SIM-PE: PRIVATE EVOLUTION (PE) WITH SIMULATORS

In this paper, we focus on DP *image* generation. The beauty of the PRIVATE EVOLUTION framework is that it isolates *DP mechanism* from *data generation backend*. In particular, any data generation backend that supports RANDOM_API and VARIATION_API can be plugged into the framework and transformed into a DP data synthesis algorithm. Therefore, **our goal is to design RANDOM_API and VARIATION_API for image simulators.**

We notice that existing popular image simulators provide different levels of access. **Some simulators are open-sourced.** Examples include KUBRIC (Greff et al., 2022), a Blender-based renderer for multi-object images/videos; 3D TEAPOT (Lin et al., 2020b; Eastwood & Williams, 2018), an OpenDR-based renderer for teapot images; and PYTHON-AVATAR (Escartín, 2021), a rule-based generator for avatars. However, the assets (e.g., 3D models) used in these renderers are often proprietary. Therefore, **many simulator works choose to release only the generated datasets without the simulator code.** Examples include the FACE SYNTHETICS (Wood et al., 2021) and DIGIFACE-1M (Bae et al., 2023) datasets, both generated using Blender-based renderers for human faces. In § 3.1 and 3.2, we discuss the design for simulators with code access and data access, respectively.

Moreover, since simulators and foundation models provide the same RANDOM_API and VARIATION_API interfaces to PE, PE can easily utilize both together. App. B discusses the methedology.

Privacy analysis. Since we only modify RANDOM_API and VARIATION_API, the privacy guarantee is exactly the same as Lin et al. (2023).

3.1 SIM-PE WITH SIMULATOR ACCESS

While different simulators have very different programming interfaces, most of them can be abstracted in the same way. Given a set of p *categorical* parameters $\xi_1, \ldots, \xi_p$ and q *numerical* parameters $\phi_1, \ldots, \phi_q$ where $\xi_i \in \Xi_i$ and $\phi_i \in \Phi_i$, the simulator $\mathcal{S}$ generates an image $\mathcal{S}(\xi_1, \ldots, \xi_p, \phi_1, \ldots, \phi_q)$. For example, for face image renders (Wood et al., 2021; Bae et al., 2023), ξ_is could be the ID of the 3D human face model and the ID of the hair style, and ϕ_is could be the angle of the face and the strength of lighting.

For RANDOM_API, we simply draw each parameter randomly from its corresponding feasible set. Specifically, we define RANDOM_API $= \mathcal{S}(\xi_1, \ldots, \xi_p, \phi_1, \ldots, \phi_q)$, where $\xi_i \sim \text{Uniform}(\Xi_i)$ and $\phi_i \sim \text{Uniform}(\Phi_i)$. Here, $\text{Uniform}(S)$ denotes drawing a sample uniformly at random from the set S.

For VARIATION_API, we generate variations by perturbing the input image parameters. For numerical parameters ϕ_i, we simply add small noise. However, for categorical parameters ξ_i, where no natural ordering exists among feasible values in Ξ_i, adding noise is not applicable. Instead, we re-draw the parameter from the entire feasible set Ξ_i with a certain probability. Formally, it is defined as VARIATION_API $(\mathcal{S}(\xi_1, \ldots, \xi_p, \phi_1, \ldots, \phi_q)) = \mathcal{S}(\xi'_1, \ldots, \xi'_p, \phi'_1, \ldots, \phi'_q)$, where $\phi'_i \sim \text{Uniform}([\phi_i - \alpha, \phi_i + \alpha] \cap \Phi_i)$ and $\xi'_i = \begin{cases} \text{Uniform}(\Xi_i), & \text{with probability } \beta \\ \xi_i, & \text{with probability } 1-\beta \end{cases}$. Here, α and β control the degree of variation. At one extreme, when $\alpha = \infty$ and $\beta = 1$, VARIATION_API completely discards the information of the input sample and reduces to RANDOM_API. Conversely, when $\alpha = \beta = 0$, VARIATION_API outputs the input sample unchanged.

3.2 SIM-PE WITH SIMULATOR-GENERATED DATA

Here, we assume that a dataset of m samples $S_{\text{sim}} = \{z_1, \ldots, z_m\}$ generated from the simulator is already given. The goal is to pick N_{syn} samples from them to construct the DP synthetic dataset S_{syn}. Before discussing our final solutions, we first discuss two straightforward approaches.

Baseline 1: Applying DP_NN_HISTOGRAM on S_{syn}. One immediate solution is to apply DP_NN_HISTOGRAM in PE (Alg. 2) by treating S_{sim} as the generated set S. In other words, each private sample votes for its nearest neighbor in S_{sim}, and the final histogram, aggregating all votes, is privatized with Gaussian noise. We then draw samples from S_{sim} according to the privatized histogram (i.e., Line 8 in Alg. 1) to obtain S_{syn}.

However, the size of the simulator-generated dataset (i.e., m) is typically very large (e.g., 1.2 million in Bae et al. (2023)), and the total amount of added Gaussian noise grows with m. This means that the resulting histogram suffers from a low signal-to-noise ratio, leading to poor fidelity in S_{syn}.

Baseline 2: Applying DP_NN_HISTOGRAM on cluster centers of S_{syn}. To improve the signal-to-noise ratio of the histogram, one solution is to have private samples vote on the cluster centers of S_{sim} instead of the raw samples. Specifically, we first cluster the samples in S_{sim} into N_{cluster} clusters with centers $\{w_1, \ldots, w_{N_{\text{cluster}}}\}$ and have private samples vote on these centers rather than individual samples in S_{sim}.[1] Since the number of bins in the histogram decreases from m to N_{cluster}, the signal-to-noise ratio improves. Following the approach of the previous baseline, we then draw N_{syn} cluster centers (with replacement) based on the histogram and randomly select a sample from each chosen cluster to construct the final S_{syn}.

However, when the total number of samples m is large, each cluster may contain a diverse set of samples, including those both close to and far from the private dataset. While DP voting on clusters improves the accuracy of the DP histogram and helps select better clusters, there remains a risk of drawing unsuitable samples from the chosen clusters.

Our approach. Our key insight is that the unavoidable trade-off between the accuracy of the DP histogram and the precision of selection (clusters vs. individual samples) arises because private samples are forced to consider all samples in S_{sim}—either directly in baseline 1 or indirectly through cluster centers in baseline 2. However, this is not necessary. If we already know that a sample z_i is far from the private dataset, then its nearest neighbors in S_{sim} are also likely to be far from the private dataset. Therefore, we can avoid wasting the privacy budget on evaluating such samples.

The iterative selection and refinement process in PE naturally aligns with this idea. For each sample z_i, we define its nearest neighbors in S_{sim} as $q_1^i, \ldots, q_m^i$, sorted by closeness, where $q_1^i = z_i$ is the closest. We define RANDOM_API as drawing a random sample from S_{sim}: $\text{RANDOM\_API} \sim \text{Uniform}\left(S_{\text{sim}}\right)$. Since we only draw N_{syn} samples (instead of m) from RANDOM_API, the DP histogram on this subset has a higher signal-to-noise ratio. In the following steps (Lines 6 to 8), samples far from the private dataset are removed, and we perform variations only on the remaining samples according to: $\text{VARIATION\_API}\left(z_i\right) = \text{Uniform}\left(\left\{q_1^i, \ldots, q_\gamma^i\right\}\right)$, thus avoiding consideration of nearest neighbors of removed samples (unless they are also nearest neighbors of retained samples). Similar to α and β and in § 3.1, the parameter γ controls the degree of variation. At one extreme, when $\gamma = m$, VARIATION_API disregards the input sample and reduces to RANDOM_API. At the other extreme, when $\gamma = 1$, VARIATION_API returns the input sample unchanged.

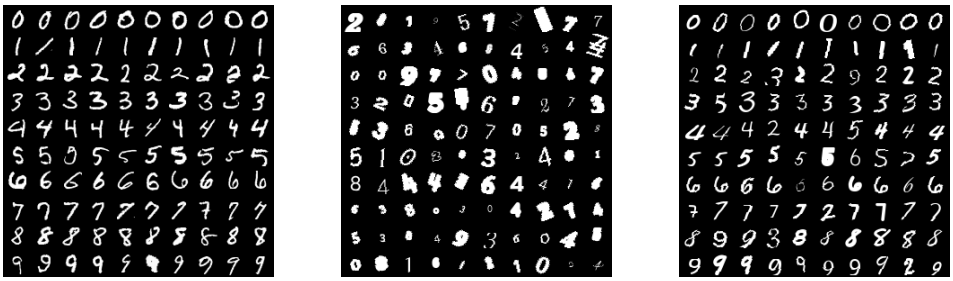
Broader applications. The proposed algorithm can be applied to any public dataset beyond simulator-generated data. In our experiments (§ 4), we focus on simulator-generated data, and we leave the exploration of broader applications for future work.

4 Experiments

Datasets. Following prior work, we use two private datasets: **(1) MNIST** (LeCun, 1998), where the image class labels are digits '0'-'9', and **(2) CelebA** (Liu et al., 2015), where the image class labels are male and female. We aim at *conditional generation* for these datasets (i.e., each generated image is associated with the class label).

Simulators. To demonstrate the general applicability of Sim-PE, we select three diverse simulators with very different implementations, including a text rendering program for MNIST, a computer graphics-based renderer for CelebA (data access only), and a rule-based avatar generator for CelebA. See App. D.1 for the details. In the main experiments, we assume the challenging setting where the class label information from the simulators is *not* available to Sim-PE; in App. B, we show that Sim-PE's performance could be even better with the class label information.

[1] Note that voting in (Lin et al., 2023) is conducted in the image embedding space. Here, w_is represent cluster centers in the embedding space, and each private sample uses its image embedding to find the nearest cluster center.



(a) Real (private) images (b) Simulator-generated (c) SIM-PE-generated ($\epsilon = 10$)

Figure 1: The real and generated images on MNIST. Each row corresponds to one class. The simulator generates images that are very different from the real ones and are from the incorrect classes. Starting from these bad images, SIM-PE can effectively guide the generation of the simulator towards high-quality images with correct classes.

Algorithm	MNIST		CelebA	
	$\epsilon = 1$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 10$
DP-MERF	80.3	81.3	81.0	81.2
DP-NTK	50.0	91.3	61.2	64.2
DP-Kernel	94.0	93.6	83.0	83.7
GS-WGAN	72.4	75.3	61.4	61.5
DP-GAN	92.4	92.7	77.9	89.2
DPDM	89.2	97.7	74.5	91.8
PDP-Diffusion	94.5	97.4	89.4	94.0
DP-LDM (SD)	78.8	94.4	84.4	89.1
DP-LDM	44.2	95.5	85.8	92.4
DP-LoRA	82.2	97.1	87.0	92.0
PrivImage	94.0	97.8	90.8	92.0
Simulator	11.6 ($\epsilon = 0$)		61.4 ($\epsilon = 0$)	
PE	27.9	32.7	70.5	74.2
SIM-PE (ours)	**89.1**	**93.6**	**80.0**	**82.5**

(a) Accuracy (%) of downstream classifiers.

Algorithm	MNIST		CelebA	
	$\epsilon = 1$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 10$
DP-MERF	113.7	106.3	176.3	147.9
DP-NTK	382.1	69.2	350.4	227.8
DP-Kernel	33.7	38.9	140.3	128.8
GS-WGAN	57.0	47.7	611.8	290.0
DP-GAN	82.3	30.3	112.5	31.7
DPDM	36.1	4.4	153.99	28.8
PDP-Diffusion	8.9	3.8	17.1	8.1
DP-LDM (SD)	31.9	18.7	46.2	24.1
DP-LDM	155.2	99.1	124.1	40.4
DP-LoRA	112.8	95.4	53.3	32.2
PrivImage	7.6	2.3	11.4	11.3
Simulator	86.2 ($\epsilon = 0$)		37.2 ($\epsilon = 0$)	
PE	48.8	45.3	**23.4**	22.0
SIM-PE (ours)	**20.7**	**9.4**	24.7	**20.8**

(b) FID of synthetic images.

Table 1: Accuracy and FID. The best between PE methods is in **bold**, and the best between all methods is underlined. "Simulator" refers to samples from the simulator's RANDOM_API. Results other than SIM-PE and Simulator are taken from Gong et al. (2025).

Metrics. We follow the evaluation settings of DPImageBench (Gong et al., 2025), a recent benchmark for DP image synthesis. Specifically, we use two metrics: **(1) FID** (Heusel et al., 2017) and **(2) the accuracy of downstream classifiers**. For (2), we employ a strict train-validation-test split and account for the privacy cost of classifier hyperparameter selection. See App. D.2 for the details.

DP parameters. Following Gong et al. (2025), we set DP parameter $\delta = 1/(N_{\text{priv}} \cdot \log N_{\text{priv}})$, where N_{priv} is the number of samples in the private dataset, and $\epsilon = 1$ or 10.

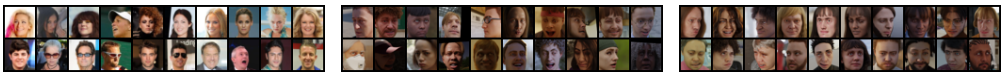
Baselines We compare SIM-PE with 12 state-of-the-art DP image synthesizers reported in Gong et al. (2025), including DP-MERF (Harder et al., 2021a), DP-NTK (Yang et al., 2023), DP-Kernel (Jiang et al., 2023), GS-WGAN (Chen et al., 2020), DP-GAN (Xie et al., 2018), DPDM (Dockhorn et al., 2023), PDP-Diffusion (Ghalebikesabi et al., 2023b), DP-LDM (Liu et al., 2024b), DP-LoRA (Tsai et al., 2024), PrivImage (Li et al., 2024), and PE with foundation models (Lin et al., 2023). Except for PE, all baselines require model training. For SIM-PE with simulator-generated data, we also compare it against the two baselines introduced in § 3.2; see App. C.2 for results.

It is important to note that this comparison is not intended to be entirely fair, as different methods leverage different prior knowledge (pre-trained models or simulators). SIM-PE should be considered as a new simulator-based benchmark, and the baseline results serve as a reference.

4.1 SIM-PE WITH SIMULATOR ACCESS

In this section, we evaluate SIM-PE with a text rendering program on MNIST dataset. The results are shown in Tables 1 and 3 and Figs. 1 and 4. The key takeaway messages are:

SIM-PE effectively guides the simulator to generate high-quality samples. As shown in Fig. 1b, without any information from the private data or guidance from SIM-PE, the simulator initially produces poor-quality images with incorrect digits, digit sizes, rotations, and stroke widths. These low-quality samples serve as the starting point for SIM-PE (via RANDOM_API). Through iterative



(a) Real (private) images (b) Simulator-generated (c) SIM-PE-generated ($\epsilon = 10$)

Figure 2: The real and generated images on CelebA. The top rows correspond to the "female" class, and the bottom rows correspond to the "male" class. The simulator generates images with incorrect classes. However, starting from these misclassified images, SIM-PE effectively selects those that better match the correct class.

refinement and private data voting, SIM-PE gradually optimizes the simulator parameters, ultimately generating high-quality MNIST samples, as illustrated in Fig. 1c.

Quantitative results in Table 1 further support this. Without private data guidance, the simulator naturally generates digits from incorrect classes, leading to a downstream classifier accuracy of only 11.6%, close to random guessing. In contrast, SIM-PE significantly improves accuracy to approximately 90%. Additionally, FID scores confirm that the images from SIM-PE are more realistic.

SIM-PE can significantly improve the performance of PE. The PE baseline (Lin et al., 2023) uses a diffusion model pre-trained on ImageNet, which primarily contains natural object images (e.g., plants, animals, cars). Since MNIST differs significantly from such data, PE, as a training-free method, struggles to generate meaningful images. Most PE-generated images lack recognizable digits (see Gong et al. (2025)), resulting in a classification accuracy of only $\sim 30\%$ (Table 1a). By leveraging a simulator better suited for this domain, SIM-PE achieves significantly better results, tripling the classification accuracy and reducing the FID score by 80% at $\epsilon = 10$.

SIM-PE achieves competitive results among state-of-the-art methods. When the foundation model or public data differs significantly from the private data, training-based baselines can still adapt the model to the private data distribution by updating its weights, whereas PE cannot. This limitation accounts for the substantial performance gap between PE and other methods. Specifically, PE records the lowest classification accuracy among all 12 methods (Table 1a). By leveraging domain-specific simulators, SIM-PE significantly narrows this gap, achieving classification accuracy within 5.4% and 4.2% of the best-performing method for $\epsilon = 1$ and $\epsilon = 10$, respectively.

4.2 SIM-PE WITH SIMULATOR-GENERATED DATA

In this section, we evaluate SIM-PE using a generated dataset from a computer graphics-based renderer on the CelebA dataset. The results (Table 1 and Fig. 2) highlight the following key takeaways:

SIM-PE effectively selects samples that better match the correct classes. Without any information from the private data, the simulator naturally generates images with incorrect class labels (Fig. 2b). Consequently, a downstream classifier can at best achieve a trivial accuracy 61.4%—equaling the fraction of the majority class (female) on the test set. On top of this noisy data, SIM-PE iteratively refines the sample selection. Ultimately, SIM-PE selects samples that better align with the target classes (Fig. 2c), leading to an accuracy improvement of up to 21.1% (Table 1a).

SIM-PE maintains the strong data quality of PE. As shown in Table 1b, SIM-PE and PE achieve similar FID scores. Unlike in the MNIST experiments (§ 4.1), where SIM-PE significantly improved over PE, the lack of substantial improvement on CelebA can be attributed to two factors. First, on CelebA, PE with foundation models already ranks 3rd among all methods in terms of FID, leaving little room for further gains. Second, in this experiment, SIM-PE is only provided with a fixed dataset generated from the simulator. As seen in Fig. 2, the simulator-generated images exhibit noticeable differences from real CelebA images, such as faces appearing larger. Since SIM-PE in this setting can only select images without modifying them, it cannot correct such discrepancies. Having access to simulator code, as in § 4.1, could potentially alleviate this issue, as SIM-PE could learn to modify the parameters that control face sizes. Another potential improvement is a hybrid approach that leverages both foundation models and simulators in PE, which we explore in App. B.

5 LIMITATIONS AND FUTURE WORK

In this paper, we demonstrate the potential of the PE framework for utilizing powerful simulators in DP image synthesis. Further extensions include:

- The approach in § 3.2 can be extended beyond simulator-generated datasets, such as public web data. This could further enhance the performance of PE and enable its application in other areas, such as pre-training data selection for private fine-tuning (Yu et al., 2023; Li et al., 2024).
- This paper focuses on image generation. SIM-PE could offer greater potential in domains like networking and systems where foundation models are rarer and simulators are more prevalent.

REFERENCES

Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

Gwangbin Bae, Martin de La Gorce, Tadas Baltrušaitis, Charlie Hewitt, Dong Chen, Julien Valentin, Roberto Cipolla, and Jingjing Shen. Digiface-1m: 1 million digital face images for face recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3526–3535, 2023.

Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Tianshi Cao, Alex Bie, Arash Vahdat, Sanja Fidler, and Karsten Kreis. Don't generate me: Training differentially private generative models with sinkhorn divergence. *Advances in Neural Information Processing Systems*, 34:12480–12492, 2021.

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*, 2020.

Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pp. 715–733. Springer, 2020.

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, et al. Differentially private diffusion models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=ZPpQk7FJXF.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pp. 265–284. Springer, 2006.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *6th International Conference on Learning Representations*, 2018.

Epic Games. Unreal engine. URL https://www.unrealengine.com.

Ibon Escartín. python avatars. https://github.com/ibonn/python_avatars, 2021.

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. *arXiv preprint arXiv:2302.13861*, 2023a.

Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, et al. Differentially private diffusion models generate useful synthetic images. *CoRR*, abs/2302.13861, 2023b.

Chen Gong, Kecen Li, Zinan Lin, and Tianhao Wang. Dpimagebench: A unified benchmark for differentially private image synthesis. 2025.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Google. Google fonts. https://github.com/google/fonts, 2022.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. 2022.

Frederik Harder, Kamil Adamczewski, and Mijung Park. DP-MERF: differentially private mean embeddings with random features for practical privacy-preserving data generation. In *AISTATS*, pp. 1819–1827, 2021a.

Frederik Harder, Kamil Adamczewski, and Mijung Park. Dp-merf: Differentially private mean embeddings with randomfeatures for practical privacy-preserving data generation. In *International conference on artificial intelligence and statistics*, pp. 1819–1827. PMLR, 2021b.

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.

Jiyan He, Xuechen Li, Da Yu, Huishuai Zhang, Janardhan Kulkarni, Yin Tat Lee, Arturs Backurs, Nenghai Yu, and Jiang Bian. Exploring the limits of differentially private deep learning with group-wise clipping. *arXiv preprint arXiv:2212.01539*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Charlie Hou, Akshat Shrivastava, Hongyuan Zhan, Rylan Conway, Trang Le, Adithya Sagar, Giulia Fanti, and Daniel Lazar. Pre-text: Training language models on private federated data in the age of llms. *arXiv preprint arXiv:2406.02958*, 2024.

Teerawat Issariyakul, Ekram Hossain, Teerawat Issariyakul, and Ekram Hossain. *Introduction to network simulator 2 (NS2)*. Springer, 2009.

Dihong Jiang, Sun Sun, and Yaoliang Yu. Functional renyi differential privacy for generative modeling. In *Advances in Neural Information Processing Systems*, 2023.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2019.

Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4551–4560, 2019.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Kecen Li, Chen Gong, Zhixiang Li, et al. PrivImage: Differentially private synthetic image generation using diffusion models with Semantic-Aware pretraining. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 4837–4854, 2024. ISBN 978-1-939133-44-1.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*, 2021.

Zinan Lin. *Data Sharing with Generative Adversarial Networks: From Theory to Practice*. PhD thesis, Carnegie Mellon University, 2022.

Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, pp. 464–483, 2020a.

Zinan Lin, Kiran Thekumparampil, Giulia Fanti, and Sewoong Oh. Infogan-cr and modelcentrality: Self-supervised model training and selection for disentangling gans. In *international conference on machine learning*, pp. 6127–6139. PMLR, 2020b.

Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, Harsha Nori, and Sergey Yekhanin. Differentially private synthetic data via foundation model APIs 1: Images. In *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*, 2023. URL https://openreview.net/forum?id=7GbfIEvoS8.

Enshu Liu, Xuefei Ning, Yu Wang, and Zinan Lin. Distilled decoding 1: One-step sampling of image auto-regressive models with flow matching. *arXiv preprint arXiv:2412.17153*, 2024a.

Michael F. Liu, Saiyue Lyu, Margarita Vinaroz, and Mijung Park. Differentially private latent diffusion models. 2024b.

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.

OpenAI. Gpt-4 technical report, 2023.

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*, pp. 10684–10695, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Yu-Lin Tsai, Yizhe Li, Zekai Chen, Po-Yu Chen, Chia-Mu Yu, Xuebin Ren, and Francois Buet-Golfouse. Differentially private fine-tuning of diffusion models. *arXiv preprint arXiv:2406.01355*, 2024.

Margarita Vinaroz, Mohammad-Amin Charusaie, Frederik Harder, Kamil Adamczewski, and Mi Jung Park. Hermite polynomial features for private data generation. In *International Conference on Machine Learning*, pp. 22300–22324. PMLR, 2022.

Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. Fake it till you make it: face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3681–3691, 2021.

Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, et al. Differentially private synthetic data via foundation model apis 2: Text. *arXiv preprint arXiv:2403.01749*, 2024.

Liyang Xie, Kaixiang Lin, and et al. Differentially private generative adversarial network. *CoRR*, abs/1802.06739, 2018. URL http://arxiv.org/abs/1802.06739.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Yilin Yang, Kamil Adamczewski, and et al. Differentially private neural tangent kernels for privacy-preserving data generation. *CoRR*, abs/2303.01687, 2023.

Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pp. 458–472, 2022.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*, 2021.

Da Yu, Sivakanth Gopi, Janardhan Kulkarni, Zinan Lin, Saurabh Naik, Tomasz Lukasz Religa, Jian Yin, and Huishuai Zhang. Selective pre-training for private fine-tuning. *arXiv preprint arXiv:2305.13865*, 2023.

Xiang Yue, Huseyin A Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Huan Sun, David Levitan, and Robert Sim. Synthetic text generation with differential privacy: A simple and practical recipe. *arXiv preprint arXiv:2210.14348*, 2022.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

A PRIVATE EVOLUTION

Alg. 1 presents the PRIVATE EVOLUTION (PE) algorithm, reproduced from Lin et al. (2023). This algorithm represents the conditional version of PE, where each generated image is associated with a class label. It can be interpreted as running the unconditional version of PE separately for each class.

Algorithm 1: PRIVATE EVOLUTION (PE)

Input: The set of private classes: C ($C = \{0\}$ if for unconditional generation)
Private samples: $S_{\text{priv}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{priv}}}$, where x_i is a sample and $y_i \in C$ is its label
Number of iterations: T
Number of generated samples: N_{syn} (assuming N_{syn} mod $|C| = 0$)
Noise multiplier for DP Nearest Neighbors Histogram: σ
Threshold for DP Nearest Neighbors Histogram: H

$S_{\text{syn}} \leftarrow \emptyset$
for $c \in C$ **do**
  $private\_samples \leftarrow \{x_i | (x_i, y_i) \in S_{\text{priv}} \text{ and } y_i = c\}$
  $S_1 \leftarrow$ RANDOM_API $(N_{\text{syn}}/|C|)$
  for $t \leftarrow 1, \ldots, T$ **do**
    $histogram_t \leftarrow$ DP_NN_HISTOGRAM $(private\_samples, S_t, \sigma, H)$ `// See Alg. 2`
    $\mathcal{P}_t \leftarrow histogram_t / \text{sum}(histogram_t)$ `// $\mathcal{P}_t$ is a distribution on $S_t$`
    $S'_t \leftarrow$ draw $N_{\text{syn}}/|C|$ samples with replacement from $\mathcal{P}_t$ `// $S'_t$ is a multiset`
    $S_{t+1} \leftarrow$ VARIATION_API (S'_t)
  $S_{\text{syn}} \leftarrow S_{\text{syn}} \cup \{(x, c) | x \in S_T\}$
return S_{syn}

Algorithm 2: DP Nearest Neighbors Histogram (DP_NN_HISTOGRAM)

Input : Private samples: S_{priv}
Generated samples: $S = \{z_i\}_{i=1}^{n}$
Noise multiplier: σ
Threshold: H
Distance function: $d(\cdot, \cdot)$
Output: DP nearest neighbors histogram on S

$histogram \leftarrow [0, \ldots, 0]$
for $x_{\text{priv}} \in S_{\text{priv}}$ **do**
  $i = \arg\min_{j \in [n]} d(x_{\text{priv}}, z_j)$
  $histogram[i] \leftarrow histogram[i] + 1$
$histogram \leftarrow histogram + \mathcal{N}(0, \sigma I_n)$ `// Add noise to ensure DP`
$histogram \leftarrow \max(histogram - H, 0)$ `// 'max', '-' are element-wise`
return $histogram$

B SIM-PE WITH BOTH SIMULATORS AND FOUNDATION MODELS

B.1 APPROACH

As discussed in § 2, simulators and foundation models complement each other across different data domains. Moreover, even within a single domain, they excel in different aspects. For example, computer graphics-based face image generation frameworks (Bae et al., 2023; Wood et al., 2021) allow controlled diversity in race, lighting, and makeup while mitigating potential biases in foundation models. However, the generated faces may appear less realistic than those produced by state-of-

(a) Real (private) images (b) Simulator-generated images (c) SIM-PE-generated images ($\epsilon = 10$)

Figure 3: The real and generated images on CelebA. The simulator is a weak rule-based avatar generator (Escartín, 2021) significantly different from the real dataset. The top rows correspond to the "female" class, and the bottom rows correspond to the "male" class. The simulator generates images with incorrect classes. SIM-PE tends to generate faces with long hair for the female class and short hair for the male class (correctly), but the generated images have mode collapse issues.

the-art foundation models. Thus, combining the strengths of both methods for DP data synthesis is highly appealing.

Fortunately, PE naturally supports this integration, as it only requires RANDOM_API and VARIATION_API, which work the same for both foundation models and simulators. While there are many ways to combine them, we explore a simple strategy: using simulators in the early PE iterations to generate diverse seed samples, then switching to foundation models in later iterations to refine details and enhance realism. As shown in § 4, this approach outperforms using either simulators or foundation models alone.

B.2 RESULTS

In this section, we examine how SIM-PE performs with weak simulators. We again use the CelebA dataset as the private data, but this time, we switch to a rule-based cartoon avatar generator (Escartín, 2021) as the simulator. As shown in Fig. 3, the avatars generated by the simulator differ significantly from the real CelebA images.

SIM-PE with weak simulators still learns useful features. From Table 2, we observe that downstream classifiers trained on SIM-PE with weak simulators achieve poor classification accuracy. However, two interesting results emerge: **(1)** Despite the significant difference between avatars and real face images, SIM-PE still captures certain characteristics of the two classes correctly. Specifically, SIM-PE tends to generate faces with long hair for the female class and short hair for the male class (Fig. 3). **(2)** Although the FID scores of SIM-PE are quite poor (Table 2), they still outperform many baselines (Table 1b). This can be explained by the fact that, as shown in Gong et al. (2025), when DP noise is high, the training of many baseline methods becomes unstable, leading to collapse. This results in face images with noisy patterns, non-face images, or significant mode collapse, particularly for DP-NTK, DP-Kernel, and GS-WGAN. In contrast, SIM-PE is training-free, and thus it avoids these issues.

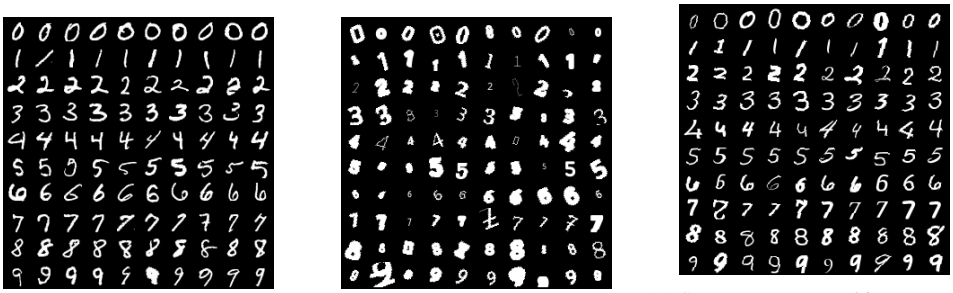
Next, we explore the feasibility of using PE with both foundation models and the weak avatar simulator (App. B). The results are shown in Table 2.

PE benefits from utilizing simulators and foundation models together. We observe that using both simulators and foundation models yields the best results in terms of both FID and classification accuracy. This result is intuitive: the foundation model, pre-trained on the diverse ImageNet dataset, has a low probability of generating a face image through RANDOM_API. While avatars are quite different from CelebA, they retain the correct image layout, such as facial boundaries, eyes, nose, etc. Using these avatars as seed samples for variation allows the foundation model to focus on images closer to real faces, rather than random, unrelated patterns.

Unlike other state-of-the-art methods that are tied to a specific data synthesizer, this result suggests that PE is a promising framework that can easily combine the strengths of multiple types of data synthesizers.

Table 2: Accuracy (%) of classifiers trained on synthetic images and FID of synthetic images on CelebA. The best results are highlighted in bold. Using a combination of both (weak) simulators and foundation models outperforms using either one alone.

Algorithm	FID ↓		Classification Acc. ↑	
	$\epsilon = 1$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 10$
PE with foundation models	23.4	22.0	70.5	74.2
PE with weak simulators (i.e., SIM-PE)	101.4	99.5	62.6	63.2
PE with both	**15.0**	**11.9**	**72.7**	**78.1**



(a) Real (private) images (b) Simulator-generated images (c) SIM-PE-generated images ($\epsilon = 10$)

Figure 4: The real and generated images on MNIST under the "**ClassAvail**" setting. Each row corresponds to one class. The simulator generates images that are very different from the real ones. Starting from these bad images, SIM-PE can effectively guide the generation of the simulator towards high-quality images that are more similar to real data.

Table 3: Accuracy (%) of classifiers trained on synthetic images and FID of synthetic images on MNIST under the "**ClassAvail**" setting. See Tables 1a and 1b for results under the "**ClassUnavail**" setting for reference.

Algorithm	FID ↓		Classification Acc. ↑	
	$\epsilon = 1$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 10$
Simulator	86.0 ($\epsilon = 0$)		92.2 ($\epsilon = 0$)	
SIM-PE	**20.7**	**8.6**	**93.9**	**95.5**

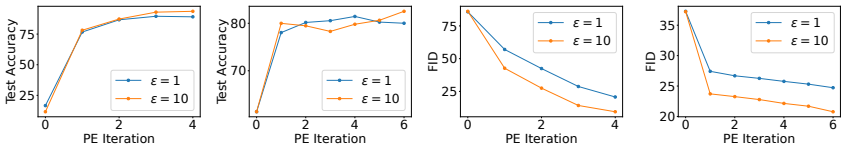
Table 4: Accuracy (%) of classifiers trained on synthetic images and FID of synthetic images on CelebA. The best results are highlighted in bold. SIM-PE outperforms the baselines in most metrics.

Algorithm	FID ↓		Classification Acc. ↑	
	$\epsilon = 1$	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 10$
DP_NN_HISTOGRAM on S_{syn}	36.2	29.3	61.5	71.9
DP_NN_HISTOGRAM on cluster centers of S_{syn}	26.4	**18.3**	74.7	77.7
SIM-PE	**24.7**	20.8	**80.0**	**82.5**

C MORE RESULTS

C.1 WHEN CLASS LABEL INFORMATION FROM THE SIMULATORS IS AVAILABLE

Class label information from the simulators can be helpful. All the above experiments are based on the **ClassUnavail** setting, where the class label information from the simulator is assumed to be unknown. However, one key advantage of using simulators over foundation models for generating synthetic data is that simulators can provide various labels for free (Wood et al., 2021; Bae et al., 2023). In our case, for MNIST, the simulators provide information on which digit the generated image represents. Following the approach in § 4, we utilize this label information, and the results are presented in Table 3 and Fig. 4. We observe that with digit information, the simulator-generated data achieve significantly higher classification accuracy (92.2%), although the FID remains low due to the generated digits exhibiting incorrect characteristics (Fig. 4b). The fact that SIM-PE outperforms the simulator in both FID and classification accuracy across all settings suggests that SIM-PE effectively incorporates private data information to enhance both data fidelity and utility, even when compared to such a strong baseline. As expected, SIM-PE under **ClassAvail** matches or surpasses the results obtained in **ClassUnavail** across all settings, suggesting the usefulness of leveraging class label information.



(a) Acc. on MNIST (b) Acc. on CelebA (c) FID on MNIST (d) FID on CelebA

Figure 5: SIM-PE's FID and accuracy generally improve over the course of the PE iterations.

C.2 VALIDATING THE DESIGN OF SIM-PE

In this section, we provide more experiments to understand and validate the design of SIM-PE.

How does SIM-PE with simulator-generated data compare to other data selection algorithms? In § 3.2, we discussed two simple alternative solutions for simulator data selection. The comparison is shown in Table 4. As we can see, SIM-PE with iterative data selection outperforms the baselines on most metrics, validating the intuition outlined in App. B. However, the clustering approach used in the second baseline still has merit, as it results in a better FID for $\epsilon = 10$. This idea is orthogonal to the design of SIM-PE and could potentially be combined for further improvement. We leave this exploration to future work.

How does SIM-PE's performance evolve across PE iterations? Fig. 5 shows that both the FID and the downstream classifier's accuracy generally improve as PE progresses. This confirms that PE's iterative data refinement process is effective when combined with simulators.

D EXPERIMENTAL DETAILS

In this section, we provide more experimental details.

D.1 SIMULATORS

To demonstrate the general applicability of SIM-PE, we select three diverse simulators with very different implementations.

(1) Text rendering program. Generating images with readable text using foundation models is a known challenge (Betker et al., 2023). Simulators can address this gap, as generating images with text through computer programs is straightforward. To illustrate this, we implement our own text rendering program, treating MNIST as the private dataset. Specifically, we use the Python PIL library to render digits as images. **The categorical parameters include:** (1) Font. We use Google Fonts (Google, 2022), which offers 3589 fonts in total. (2) Text. The text consists of digits '0' - '9'. Although we restrict the text to digits, the digit label is not provided to SIM-PE, which must learn and select the correct digits for each MNIST class itself. **The numerical parameters include:** (1) Font size, ranging from 10 to 29. (2) Stroke width, ranging from 0 to 2. (3) Digit rotation degree, ranging from $-30°$ to $30°$. We set the feasible sets of these parameters to be large enough so that the random samples differ significantly from MNIST (see Fig. 1b).

(2) Computer graphics-based renderer for face images. Computer graphics-based rendering is widely used in real-world applications such as game development, cartoons, and movie production. This experiment aims to assess whether these advanced techniques can be adapted for DP synthetic image generation via SIM-PE. We use CelebA as the private dataset and a Blender-based face image renderer from Bae et al. (2023) as the API. Since the source code for their renderer is not publicly available, we apply our data-based algorithm from § 3.2 on their released dataset of 1.2 million face images. It is important to note that this renderer may not necessarily represent the state-of-the-art. As visualized in Fig. 2b, the generated faces exhibit various unnatural artifacts and appear less realistic than images produced by state-of-the-art generative models (e.g., Rombach et al. (2022)). Therefore, this experiment serves as a preliminary study, and the results could potentially improve with more advanced rendering techniques.

(3) Rule-based avatar generator. We further investigate whether SIM-PE remains effective when the simulator's data significantly differs from the private dataset. We use CelebA as the private

dataset and a rule-based avatar generator (Escartín, 2021) as the API. This simulator has 16 categorical parameters that control attributes of the avatar including eyes, noses, background colors, skin colors, etc. As visualized in Fig. 3b, the generated avatars have a cartoon-like appearance and lack fine-grained details. This contrasts sharply with CelebA images, which consist of real human face photographs.

Class label information from the simulators. For simulator 1, the target class label (i.e., the digit) is fully controlled by one parameter. For simulators 2 and 3, the target class label (i.e., the gender) is not directly controlled by any parameter, but could potentially be obtained by an external image gender classifier. One benefit of using domain-specific simulators is that we can potentially use the class label information to enhance data quality. To get a more comprehensive understanding of SIM-PE, we consider two settings: **(1) Class label information is unavailable (abbreviated as "ClassUnavail").** We artificially make the problem more challenging by assuming that the class label information is *not* available. Therefore, SIM-PE has to learn to synthesize images with the correct class by itself. **(2) Class label information is available (abbreviated as "ClassAvail").** On MNIST, we further test how SIM-PE can be improved if the class label information is available. In this case, the RANDOM_API and VARIATION_API (§ 3.1) are restricted to draw parameters from the corresponding class (i.e., the digit is set to the target class).

D.2 METRICS AND EVALUATION PIPELINES

We follow the evaluation settings of DPImageBench (Gong et al., 2025), a recent benchmark for DP image synthesis. Specifically, we use two metrics: **(1) FID** (Heusel et al., 2017) as a quality metric and **(2) the accuracy of downstream classifiers** as a utility metric. Specifically, we use the conditional version of PE (App. A), so that each generated images are associated with the class labels (i.e., '0'-'9' digits in MNIST, male vs. female in CelebA). These class labels are the targets for training the classifiers. We employ a strict train-validation-test split and account for the privacy cost of classifier hyperparameter selection. Specifically, we divide the private dataset into disjoint training and validation sets. We then run SIM-PE on the training set to generate synthetic data. Next, we train three classifiers—ResNet (He et al., 2016), WideResNet (Zagoruyko & Komodakis, 2016), and ResNeXt (Xie et al., 2017)—on the synthetic data and evaluate their accuracy on the validation set. Since the validation set is part of the private data, we use the Report Noisy Max algorithm (Dwork et al., 2014) to select the best classifier checkpoint across all epochs of all three classifiers. Finally, we report the accuracy of this classifier on the test set. This procedure ensures that the reported accuracy is not inflated due to train-test overlap or DP violations in classifier hyperparameter tuning.

D.3 MNIST WITH TEXT RENDERING PROGRAM

Tables 5 and 6 show the list of the parameters and their associated feasible sets and variation degrees in the MNIST with Text Rendering Program experiments. The total number of PE iterations is 4.

Categorical Parameter (ξ)	Feasible Set (Ξ)	Variation Degrees (β) Across PE Iterations
Font	1 - 3589	0.8, 0.4, 0.2, 0.0
Text	'0' - '9'	0, 0, 0, 0

Table 5: The configurations of the categorical parameters in MNIST with Text Rendering Program experiments.

Numerical Parameter (ϕ)	Feasible Set (Φ)	Variation Degrees (α) Across PE Iterations
Font size	[10, 30]	5, 4, 3, 2
Font rotation	[-30, 30]	9, 7, 5, 3
Stroke width	[0, 2]	1, 1, 0, 0

Table 6: The configurations of the numerical parameters in MNIST with Text Rendering Program experiments.

D.4 CelebA with Generated Images from Computer Graphics-based Render

The variation degrees γ across PE iterations are [1000, 500, 200, 100, 50, 20]. The total number of PE iterations is 6.

D.5 CelebA with Rule-based Avatar Generator

The full list of the categorical parameters are

- Style
- Background color
- Top
- Hat color
- Eyebrows
- Eyes
- Nose
- Mouth
- Facial hair
- Skin color
- Hair color
- Facial hair color
- Accessory
- Clothing
- Clothing color
- Shirt graphic

These are taken from the input parameters to the library (Escartín, 2021). The variation degrees β across PE iterations are [0.8, 0.6, 0.4, 0.2, 0.1, 0.08, 0.06]. There is no numerical parameter. The total number of PE iterations is 7.

For the experiments with both foundation models and the simulator, we use a total of 5 PE iterations so as to be consistent with the setting in Gong et al. (2025). For the RANDOM_API and the first PE iteration, we use the simulator ($\beta = 0.8$). For the next 4 PE iterations, we use the same foundation model as in Lin et al. (2023) with variation degrees [96, 94, 92, 90].