EFFICIENT EXTRACTION ATTACKS FOR UNCONDITIONAL DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models (DMs) have become one of the most powerful image generation models. However, the data privacy risks associated with them have not been fully revealed. The data leakage problem for unconditional DMs trained on images of human faces is especially concerning. In this paper, we ask: *"Is it possible to extract training data efficiently from the DMs pretrained on human face datasets?"* To answer the question, we first formulate the task of *efficient extraction attacks* in the context of human face images. We first show efficiency of extraction attacks depends on both the *speed of generation* and *per-sample hit probability*. By testing mainstream sampling acceleration methods including SDE and ODE solvers, we validate that faster samplers are indeed more efficient in extracting training data, but ODE solvers are still limited by the low per-sample hit probability. Although the SDE solvers are computationally more intensive per image, they are five times more likely to extract a training image with each generation, making them much more efficient than the ODE counterparts. This suggests that hit probability is a more critical determinant of efficiency than generation speed alone. To fundamentally boost sampling efficiency by increasing the hitting rate, we propose a Generation-Filtering-Resampling framework that adapts the distribution of initial noise to concentrate on the high-probability regions in the latent space. Experiments on both a pixel-space DM and a latent-space DM trained on CelebA-HQ and FFHQ validate the efficiency improvement of our method, which is at least twice as efficient as the existing methods in most cases.

1 INTRODUCTION

The diffusion model (DM) Sohl-Dickstein et al. (2015); Ho et al. (2020) is a kind of deep generative model inspired by non-equilibrium statistical physics. Although originally proposed for image generation Sohl-Dickstein et al. (2015), DM has been widely applied to other tasks such as video generation Ho et al. (2022); Wang et al. (2024); Jeong et al. (2025) and 3D generation Yi et al. (2024); Huang et al. (2024). Unconditional DM, which doesn't accept any signal as input conditions, is the foundation for all other kinds of DMs.

The recent development of unconditional DMs Dhariwal & Nichol (2021); Rombach et al. (2022) raises concerns about training data leakage Somepalli et al. (2023a). This is particularly concerning for DMs trained on sensitive data, such as human faces. In this paper, we study a typical scenario where an unconditional DM pretrained on a private face dataset is uploaded to the Internet, and the attacker, who has access only to the weights, tries to extract the training data from it.

As shown in Fig. 1, extraction attacks are a kind of inversion attack that tries to generate training data from the pretrained model Carlini et al. (2023). For the DMs trained on human faces, as the privacy of the face is what really concerns people, we propose a new criterion that defines extraction based on face similarity. If the face similarity between the generated image A and the training image B is larger than a pre-defined threshold τ_e, we say B is extracted by a successful generation A.

Early attempts are mostly based on a brute-force basis, one of which runs the model for over 100 million times (more than 6,000 hours) for a tiny part of the training set with merely 94 images Carlini et al. (2023). This inefficiency conceals the actual privacy risk posed by this attack. The problem, therefore, is not whether extraction is *possible* in principle, but whether it can be made *efficient* enough to constitute a practical risk. In this paper, we answer the following question: **"Is it**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

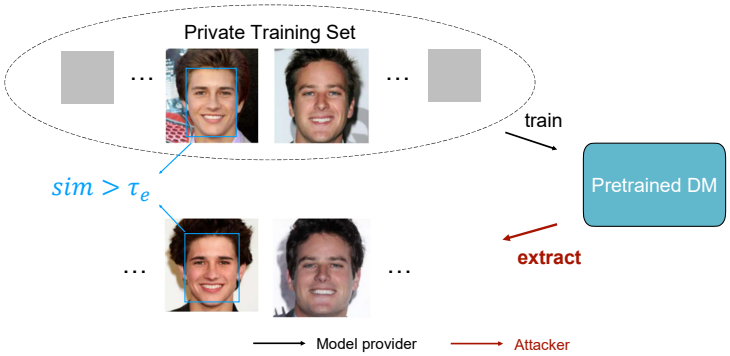


Figure 1: Diagram of extraction attacks. Extraction is defined by cosine similarity between two face features above τ_e. The τ_e is set to 0.65 in this case.

possible to extract training data efficiently from those pretrained unconditional DMs?". Since efficiency is our prior concern, we study a new problem of how to extract training data from an unconditional DM efficiently, which we term **efficient extraction attacks**. As far as we know, this is the first work discussing this topic.

Through our analysis, we find that the efficiency of the extraction is primarily influenced by the speed of generation and the per-sample hit probability (the probability of a generation extracting a training example). To see how these two factors affect extraction efficiency, we first compare common training-free samplers Song et al. (2020); Lu et al. (2022; 2025) on a DM. We observe that accelerating generation improves extraction efficiency as expected. However, for ODE solvers, the per-sample hit probability is still low across different numbers of steps. Surprisingly, SDE solvers — typically considered less efficient than their ODE counterparts for image generation — reach a much higher hit probability and turn out to be the most efficient samplers for extraction. The reason could be: the per-step noise injection in SDE samplers, combined with the score gradient, pulls trajectories back toward the high-density regions where memorized training images concentrate. On the contrary, deterministic ODE samplers, where the trajectory is completely decided by score gradient, are more likely to make the latent stuck in a local maxima. Our analysis reveals that the stochasticity in the generation process is the more important factor for efficient extraction attacks.

Though SDE solvers are proved to increase the per-sample hit probability, they still sample the initial noise from the standard Gaussian, which is not specifically designed for extracting training data. We believe that the per-sample hit probability is limited by this standard Gaussian and sampling the initial noise from a better distribution can further increase it. For that purpose, we propose a **Generation-Filtering-Resampling** framework on top of an SDE-DPM-Solver Lu et al. (2025): (i) generate an initial pool of images with the SDE solver (Generation); (ii) select those that are likely to lie near training images (Filtering); and (iii) fit a Gaussian Mixture Model on these survivors at an intermediate noise level and resample new initial latents from it (Resampling). By sampling from a distribution close to that of the training set, we can enlarge the per-sample hit probability. Through extensive experiments, we empirically validate the effectiveness of our approach on two kinds of DMs trained on CelebA-HQ Karras et al. (2018) and FFHQ Karras et al. (2019) datasets.

In summary, our contributions are as follows:

1. We provide an appropriate face-similarity-based definition of extraction in the human-face setting, and formulate the task of **efficient extraction attacks** — which has been largely overlooked in existing literature.
2. We identify that SDE solvers, despite their reputation in image generation, are the most efficient samplers for extraction. Moreover, we find that *stochasticity in the generation process is the dominant factor for efficient extraction attacks*. We believe that it makes the latent less likely to get stuck in the local probability maxima.
3. We propose the **Generation-Filtering-Resampling** framework to learn a new distribution of initial noise that favors extraction. By concentrating sampling on high-probability regions it further improves over the SDE solvers.
4. Experiments on both pixel-space and latent-space DMs trained on CelebA-HQ and FFHQ validate that our method is at least twice as efficient as existing baselines in most settings.

2 Related Works

2.1 Diffusion Models

Diffusion Model. DMs can be defined as a kind of deep generative model that bridges the data distribution and a predefined distribution through a diffusion process and a denoising process Ho et al. (2020); Sohl-Dickstein et al. (2015); Song & Ermon (2019). Although they can generate high-quality images Dhariwal & Nichol (2021), their inference time cost is too high.

Sampling Acceleration. A line of work focuses on accelerating the sampling process of diffusion models (DMs) to make them efficient Song et al. (2023); Xue et al. (2024); Salimans & Ho (2022); Zhou et al. (2024); Lu et al. (2022; 2025). To tackle the conflict between generation quality and inference efficiency, some try distilling a diffusion model to reduce the inference steps needed for sampling Salimans & Ho (2022); Song et al. (2023); Luo et al. (2023). However, these methods need optimization or training before sampling, which brings much additional time cost. Lu et al. (2022; 2025) design a series of plug-and-play schedulers, DPM-Sovler and DPM-Sovler++, that can reduce the timesteps without affecting generation quality. These methods can be directly used to make extraction attacks more efficient, but as they are not designed for extraction attacks, they still need to go through a large number of tries to extract one image. For our method, the speed of sampling is certainly important, but it is also important to increase the per-sample hit probability.

2.2 Memorization and Extraction Attacks in Diffusion Models

Some works point out and try to understand the memorization phenomenon of DMs Somepalli et al. (2023a;b); Gu et al. (2023); Bonnaire et al. (2025). The existence of this memorization phenomenon raises concerns about data leakage in DMs. Some recent studies revealed the risk of training data leakage by performing Membership Inference Attacks (MIAs) on DMs Matsumoto et al. (2023); Duan et al. (2023); Zhai et al. (2024); Dubiński et al. (2024). However, MIAs that classify a testing example as a member or a non-member do not fully reveal the potential privacy risk in DMs.

Extraction attacks, which aim at directly recovering examples from the training set, are more powerful in breaching data privacy. Carlini et al. (2023) firstly defined the extraction attacks in DMs and adopted a naive DDPM scheduler Ho et al. (2020)to extract training data from Stable Diffusion Rombach et al. (2022). Wu et al. (2025) leveraged the discrepancy between the pretrained DM and the finetuned one to extract images used in the finetuning stage. Chen et al. (2024) focuses on unconditional DMs, but it requires a pretrained classifier for guidance, which is typically not available for human face datasets. Our method differs from all these methods in that we focus on the efficiency problem of extraction attacks.

3 Threat Model and Metrics

3.1 Threat Model

In this paper, we focus on the problem of efficient extraction attack. This task involves extracting training images as quickly as possible from an unconditional DM trained on human faces. We define this problem from the following three aspects: (1) victim models (that we apply attacks on), (2) attacker, and (3) definition of extraction.

Victim Models. Unconditional diffusion models are trained on a training set $D_t = \{y_1, y_2, ..., y_n\}$ where each y_i is a training image. At the inference stage, unconditional DM does not accept any kind of condition as additional input. We especially focus on the ones trained on private face datasets and uploaded to the Internet.

Attacker. We follow a typical white-box setting Zhang et al. (2020) where the attackers have full access to all model parameters, source code, and the model architecture, but they don't know what the training data is.

Definition of extraction. Existing work Carlini et al. (2023) defines extraction as generating an image that is almost identical to one in the training set. In practice, the l_2 distance in pixel space is usually used to measure the similarity between two images. If this distance between two images is small enough, then they are deemed as "almost identical". However, in the context of human face images, this criterion can be inaccurate. If two images share the same face but differ in the

backgrounds, as shown in the third column of Fig 1, the l_2 distance between them can be abnormally large. To offer an appropriate measure, we propose to use face similarity as the distance instead. Specifically, we employ a pretrained face detector f [1] to detect the face in the generated image x_i and the training image y_i, and then encode them into features, respectively. After that, we calculate the cosine similarity $sim(f(x_i), f(y_j))$ between these features and compare it to a threshold τ_e. If $sim(f(x_i), f(y_j)) > \tau_e$, then x_i is almost identical to y_j and we say x_i extracts y_i. The benefit of this new definition lies in that the result is not easily affected by factors like backgrounds or lighting.

3.2 METRICS

Since the attacker's goal is to extract images from D_t as quickly as possible, we use the average number of extractions per hour as the metric here. In practice, for each generated image x_i, we first find the closest training example $y_i^o = argmax_{y_j \in D_t} sim(f(x_i), f(y_j))$ in the face similarity sense. Then we test whether the face similarity between x_i and y_i^o is larger than the threshold τ_e. If $sim(f(x_i), f(y_j^o)) > \tau_e$, we say x_i is a successful generation and y_i^o is extracted.

4 METHOD

In this section, we present the Generation-Filtering-Resampling (GFR) framework for efficient extraction attacks. For the set of k generated images $G = \{x_1, x_2, ..., x_k\}$, we define the set of extracted images $E(G) = \bigcup_{i=1}^k E(x_i)$, where $E(x_i)$ is the set of the image extracted by x_i:

$$E(x_i) = \begin{cases} \arg\max_{y_j \in D_t} \text{sim}(f(x_i), f(y_j)), & \text{if} \quad \max_{y_j \in D_t} \text{sim}(f(x_i), f(y_j)) > \tau_e \\ \emptyset, & \text{otherwise.} \end{cases} \tag{1}$$

Now the total number of extractions in h hour(s) is $|E(G)| = |\bigcup_{i=1}^k E(x_i)|$, and the average number of extractions per hour is $\frac{|E(G)|}{h}$. We define the probability of a generated image extracting a training image (the per-sample hit probability) as

$$\bar{p} = \frac{|E(G)|}{k} = \frac{|\bigcup_{i=1}^k E(x_i)|}{k}. \tag{2}$$

Then the average number of extractions per hour can factorize as

$$\frac{|E(G)|}{h}| = \frac{k}{h}\bar{p}. \tag{3}$$

where k/h is the average number of generations per hour. It is obvious that the efficiency of extraction attacks is decided by two factors: the speed of generation and the probability of a generated image extracting a training example. In Sec. 4.1, we discuss how these two factors affect the efficiency of the extraction attacks by examining some of the training-free sampling acceleration methods Song et al. (2020); Lu et al. (2022; 2025), and reveal that the stochasticity in the generation process is the key to better efficiency. In Sec. 4.2, we present the GFR framework to further increase $\bar{p}$.

4.1 STOCHASTICITY IN THE GENERATION PROCESS IS THE KEY

It is suggested in Eq. 3 that if we can generate more images per hour, then we can extract more images per hour. From this point of view, it seems natural for sampling acceleration methods to be used here. So, in our preliminary experiments, we compare training-free samplers including DDPM (Ho et al., 2020), DDIM (Song et al., 2020), DPM-solver (Lu et al., 2022), and its SDE counterpart (Lu et al., 2025). We use an unconditional LDM (Rombach et al., 2022) trained on CelebA-HQ as the victim model and apply these samplers to generate 100,000 images each. We then evaluate the total number of extractions and the average number of extractions per hour. Note that since the number of generation is the same for all methods, the total number of extractions actually represents the per-sample hit probability.

From Tab.1, we make the following three observations: (1) **For ODE solvers, the number of inference steps or the choice of solvers has little impact on the probability $\bar{p}$.** With the same initial noises, the total number of extractions for all ODE solvers is almost identical (rows 1, 2, 5

[1] Model from https://github.com/deepinsight/insightface.

Table 1: Preliminary experiments on sampling acceleration methods. We compare the number of extractions, the time cost, the average number of extractions per hour, and FID here.

Sampler	Step	Ext. ↑	Time ↓	Ext. per hour ↑	FID ↓
DDIM	100	15	6h 45min	2.22	25.13
DDIM	200	15	14h 20min	1.04	24.44
DDPM	100	**88**	6h 40min	13.2	28.89
DDPM	200	84	14h 10min	6.21	26.18
DPM-Solver	20	13	**2h 42min**	4.81	25.34
DPM-Solver	60	14	4h 13min	3.32	23.79
SDE-DPM-Solver	20	0	2h 44min	0	322.85
SDE-DPM-Solver	60	78	4h 19min	**18.06**	**22.98**

and 6). (2) **From the perspective of time cost, a faster sampler makes extraction attacks more efficient as expected.** Among the 100-step DDIM, the 200-step DDIM, the 60-step DPM-Solver and the 20-step DPM Solver, the 20-step DPM Solver takes the least time, and it is the most efficient one. This shows that increasing the $\frac{k}{h}$ can indeed make extraction attacks more efficient. (3) **From the perspective of the total number of extractions, stochasticity in the generation process is the key to a larger $\bar{p}$.** Among all the methods, the total number of extractions of the SDE solvers is usually much larger than that of the ODE solvers. (An exception is that the 20-step SDE-DPM-Solver extracts nothing, which can be attributed to its terrible generation quality.) Comparing the SDE solvers to the corresponding ODE solvers with the same steps, we can see that a larger $\bar{p}$ typically leads to a better efficiency.

Note that SDE solvers typically require more steps to achieve satisfactory image quality (Karras et al., 2022) and are thus considered less efficient than ODE solvers. However, the opposite is true in the context of our efficient extraction attacks. We observe that the 60-step SDE-DPM-Solver is much more efficient than the 20-step DPM-Solver, which implies that stochasticity in the generation process could be a more important factor here. To understand that, we analyze the difference between extraction attacks and normal generations. Each denoising step of DM can be viewed as moving the latent towards the high-probability regions Song et al. (2021). For normal generations, reaching the local maxima within a small area is enough for good generation quality. For extraction attacks, this is far from sufficient. Previous study (Carlini et al., 2023) proves that the reproduced training images are more likely to be among the duplicated generations, implying that training images mostly lie in some small regions of real high probability. As a result, to make an extraction, the latent has to reach these regions. For ODE solvers, updates of the latent are dominated by the score function, so when it gets near a local probability maxima, it is highly probable to get stuck there. However, for SDE solvers, the noise added in step can also affect the update of the latent, making it more likely to find its way out.

In summary, the sampling acceleration methods that focus on increasing $\frac{k}{h}$ can indeed make extraction attacks more efficient. However, we surprisingly find that stochasticity in the generation process, which affects $\bar{p}$, is an even more important factor that we should consider. We find that the 60-step SDE-DPM-Solver that considers both the generation speed and the probability $\bar{p}$ is the current best-performing sampler among them.

4.2 The Generation-Filtering-Resampling Framework

From Sec. 4.1, we realize the key factor $\bar{p}$ and the importance of the stochasticity. Digging into that, we find that, for SDE solvers, the initial noise is sampled from the standard Gaussian, which is not designed for extracting the training data. We believe that the probability $\bar{p}$ of the SDE solver is still limited by this standard Gaussian. Sampling the initial noises from a better distribution can increase our likelihood of extraction. Ideally, if we know the distribution $q^\star$ of the training images, sampling from $q^\star$ will maximize the probability $\bar{p}$. However, since we don't know what training images look like, it is impossible to sample from that distribution. We notice that the clique-finding method proposed by Carlini et al. (2023) can find replicas of training images at a high precision in this case. Inspired by this, we realize that these replicas can be used to learn a distribution $\hat{q}$ that approximates the desired distribution $q^\star$.

We design a Generation-Filtering-Resampling framework for efficient extraction attacks as shown in Fig 2. In the first stage, k_1 images are generated with the existing methods. The goal here is to

get as many extractions as possible among these k_1 generated images, so that we can collect more successful generations (the ones that are almost identical to some training images) to learn their distribution. From the experiments in Sec. 4.1, the 60-step SDE-DPM-solver Lu et al. (2025) is our best choice. The second stage is to select those images that are more likely to be replicas of the training images from those generations, which we will illustrate in Sec. 4.2.1. The third stage to be introduced in Sec 4.2.2 is to model the sub-distribution of those selected images and sample another k_2 images from the learned distribution.

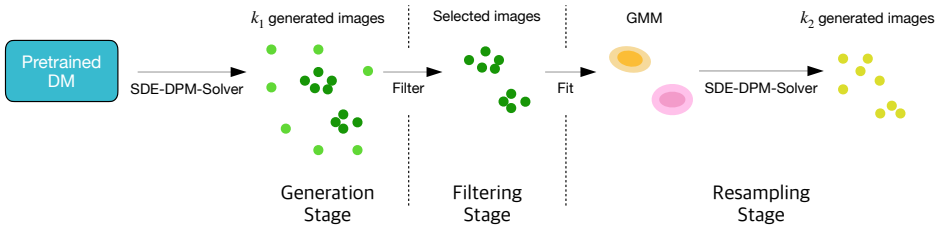


Figure 2: The Generation-Filtering-Resampling framework.

4.2.1 The Filtering Stage

The goal of the second stage is to find those replicas of training images among all generated images from stage one. For our clique-finding method, we first construct a graph where the nodes are the generated images, and calculate the similarity between each pair of nodes. If the similarity is above a threshold τ_f, then we connect these 2 nodes with an edge. After that, we search for all cliques of size larger than m (where every pair of nodes is connected). Finally, by taking the union of all these cliques, we get all the selected images for the next stage. Note that our filtering method is different from (Carlini et al., 2023) in that they use distances in pixel-space to decide if 2 nodes can be connected, while we use the face similarity here as the more appropriate metric here. This process is actually selecting the most duplicated images from all the generated images. It is corroborated that replicas of the training images are more likely to be among them (Carlini et al., 2023; Wu et al., 2025). The role of this phase is therefore one of *condensation*: by discarding low-probability candidates, we lift the proportion of true successful generations from far below 1% to a regime where a parametric model can be reliably fit on top.

4.2.2 The Resampling Stage

In the third stage, we fit a parametric proposal $\hat{q}$ to the selected images and then draw new initial latents from it.

Modeling the distribution. An intuitive try is to fit a single Gaussian. However, $q^\star$ inherits multimodal structure from the memorized training set, and a single Gaussian is too coarse to capture it, leading to a low probability $\bar{p}$. We instead use a Gaussian Mixture Model (GMM), whose c components can adapt to the cluster structure of the selected images. Specifically, we put the selected images as the set $\hat{G} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_{n_f}\}$, where $\hat{x}_i$ is a selected image. For each $\hat{x}_i$, we first obtain the latent $\hat{z}_i$ at $t^\star = 0.7T$ timestep (for a T-step DM) by reversely applying a DPM-Solver. Once it is done, we can fit a c-component GMM with these latents $\{\hat{z}_1, \hat{z}_2, ..., \hat{z}_{n_f}\}$ to get $\hat{q}$. The choice of $t^\star$ reflects the trade-off between the generation speed k/h and the probability $\bar{p}$ of making an extraction: larger $t^\star$ makes it easier to accurately model the distribution of $\{\hat{z}_1, \hat{z}_2, ..., \hat{z}_{n_f}\}$ (which is closer to a standard Gaussian); smaller $t^\star$ reduces the steps needed to denoise a latent from $t^\star$ to 0 timestep.

Sampling from the new distribution. Now that we have the $\hat{q}$, we can sample from it and get new latent at $t^\star$ timestep. Then all we have to do is denoise them with the SDE-DPM-Solver(++) used in Sec. 4.1. Note that since our denoising here starts at $t^\star = 0.7T$, the number of denoising steps here is only 70% of the original one.

5 EXPERIMENTS

5.1 IMPLEMENTATION DETAILS

To validate the effectiveness of our method, we consider two LDMs Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018) and FFHQ Karras et al. (2019) and a pixel-space DDPM Ho et al. (2020) trained on CelebA-HQ Karras et al. (2018) as our victim models. The face similarity threshold τ_e used in defining extraction is set to 0.65, which is also used to find the matching training images in Fig. 1. For both our method and the comparison ones, we generate a total of 200,000 images in all settings. Experiment results on a non-face dataset are shown in Appendix B.

The samplers used in the first stage and third stage are a 60-step SDE-DPM-Solver Lu et al. (2025) with order 2 in most cases, except for the DDPM experiment, where we use SDE-DPM-Solver++ Lu et al. (2025). The reason for such a minor difference lies in that the generation quality of the 60-step SDE-DPM-solver in pixel-space DDPM is far from satisfactory (77.97), so we turn to SDE-DPM-Solver++.

5.2 MAIN RESULTS

To show the effectiveness of our method, we conduct experiments to compare our method with the DDPM sampler used in Carlini et al. (2023), the DPM-solver Lu et al. (2022), the SDE-DPM-solver Lu et al. (2025) and SIDE-C Chen et al. (2024). The one used in Wu et al. (2025), which is designed for a pretraining-finetuning framework, does not fit here.

Table 2: Comparing our method with existing methods on three different DMs, including two LDMs trained on CelebA-HQ and FFHQ separately, and a DDPM trained on CelebA-HQ. Results with * are conducted on (SDE-)DPM-Solver++ Lu et al. (2025), which predicts data instead of noise.

Method	LDM on FFHQ		LDM on CelebA-HQ		DDPM on CelebA-HQ	
	Extr. / h ↑	Time ↓	Extr. / h ↑	Time ↓	Extr. / h ↑	Time ↓
1000-step DDPM sampler	1.26	104.4h	1.6	111.2h	0.12	249.3h
SIDE-C	0.68	27.95h	0.66	28.58	0.25	48.55h
60-step DPM-Solver	5.92	**7.6h**	4.35	**7.82h**	0^*	20.38h*
60-step SDE-DPM-Solver	9.73	8.43h	15.11	8.47h	0.81^*	19.83h*
Ours	**23.14**	8.08h	**28.78**	8.2h	**2.7***	**19.3h***

As shown in Table 2, for LDM models, the classical 1000-step DDPM sampler extracts only 1.2 and 1.6 samples per hour on two datasets respectively, which makes it impractical for face extraction in the real world. This demonstrates that extraction efficiency is a critical bottleneck for extracting attacks. We notice that SIDE-C, which is only designed for extracting more training data, is of modest efficiency. We attribute it to the large additional time cost in training and the ineffectiveness of the classifier in the context of human faces. By applying the acceleration methods, the efficiency of LDM models is improved by more than 4 times. The improvement is especially large when using the SDE-DPM-Solver, which validates the observation in 4.1 that stochasticity in the generation process is the key to a more efficient extraction.

We also conduct experiments on the DDPM model. It is shown in the table that for all (accelerated) samplers, the extraction efficiency is far from satisfactory. Specifically, the full step DDPM sampler only extracts 0.12 valid faces per hour. Constrained by the aforementioned issue in ODE samplers, the 60-step DPM solver almost fails to extract any valid samples (approx. 0 extr./h), and when it comes to its SDE counterpart, the extraction rate rises to 0.81.

With the GFR framework, we manage to boost the extraction efficiency of SDE-DPM-Solver by at least 2 times in most cases, which proves the effectiveness of our method for both kinds of DMs.

5.3 EFFICIENCY ANALYSIS

In this section, we show the effectiveness of components of the proposed Generation-Filtering-Resampling framework. As discussed in Section 4.2, our three-stage framework introduces additional time overhead beyond pure sampling. These additional time costs come from 3 aspects: (1)

getting the face features of each image in the Filtering stage, (2) running the clique-finding method in the Filtering stage, and (3) fitting a GMM in the Resampling stage. Despite the introduction of extra time cost, our framework does come with following benefits: (a) sampling from the new distribution enlarges the frequency of extraction (enlarging $\bar{p}$), (b) by modeling the distribution at 0.7T timestep (for a T-step DM), we can reduce the time cost of sampling by 30% theoretically (enlarging k/h).

To determine which factors play a dominant role, we conducted experiments on the LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018) and the results are shown in Table 3. We choose the 60-step SDE-DPM-Solver (denoted as "60-step Solver" in the table) as the baseline and use it to generate a total of 200,000 images. For our method, we first generate 100,000 images with the same SDE-DPM-Solver in the Generation stage. Then, in the Filtering stage, we first get the face features of these images (denoted as "Encoding") and then perform clique-finding on these features. At the last Resampling stage, we first fit the GMM model with the selected images (denoted as "GMM Training") and then use a 42-step SDE-DPM-Solver to sample 100,000 images from the GMM (denoted as "42-step Solver"). Note that there are duplicated extractions in the images generated by the Generation stage and the Resampling stage, so we only count the number of new extractions (per hour) for the third stage.

As shown in Table 3, though our framework brings a total of 17 minutes of additional time cost (10 minutes for encoding, 1 minute for Clique Finding, and 6 minutes for GMM training), we managed to extract 43.89 new images per hour in the Resampling stage, which is far more efficient than the baseline. This can be attributed to the greater generation speed k/h and larger probability $\bar{p}$ (more extractions per 100,000 generations). Putting all three phases together, our method not only completes the generation of 200,000 images in a shorter time (8h 16min, 12 minutes faster) than the baseline but also manages to extract more training examples (236 extractions, 2x more than the baseline). This shows that our method is more efficient than the compared baselines.

Table 3: Efficiency analysis conducted on LDM Rombach et al. (2022) trained on CelebA-HQKarras et al. (2018). The number of generations is listed in the third column. The number of (new) extractions and the number of (new) extractions per hour are listed in the fourth and fifth columns, respectively. For the third stage, we only count the number of new extractions (per hour).

Phase	Operation	Gen	Extr. ↑	Extr./h ↑	Time Cost ↓
Baseline	60-step Solver	200k	128	15.12	8h 28min
1-st	60-step Solver	100k	78	18.06	4h 19min
2-nd	Encoding		-		10min
	Clique Finding		-		1min
3-rd	GMM Training		-		6min
	42-step Solver	100k	158	43.89	3h 36min
Combined		200k	**236**	**28.78**	**8h 12min**

5.4 Ablation Studies

In this subsection, we conduct experiments with different hyperparameters to show the impact of these parameters on the final performance. We adopt the LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018) as the victim model. We follow the setting of Sec .5.3 except for that we only generate 40,000 images in the resampling stage for evaluation. Since the time cost is almost identical for each set of hyperparameters, we only evaluate the number of new extractions (the new ones that are not counted in the first stage) and the FID of the generated images in the Resampling phase.

The number of components in GMM. The number of components c in GMM is actually the number of Gaussian sub-distributions, and it is typically set to be smaller than the number of points (the number of selected images in our case). Note that a GMM with only 1 component is actually a simple Gaussian, which gets the fewest new extractions across different τ_f, indicating that it can be quite inaccurate to model the latents with a single Gaussian. Meanwhile, it seems favorable to use a mixture of more Gaussian sub-distributions for a better efficiency. We believe the larger number of components can benefit extractions from 2 aspects: (1) more accurate modeling of the distribution makes $\bar{p}$ larger; (2) sampling from a GMM that covers a larger area in the latent space makes more

Table 4: Ablation studies on the filtering threshold τ_f and the number of components c in GMM. Experiments are conducted on an LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018).

τ_f	0.45				0.4				0.35			
c	1	103	128	160	1	384	448	512	1	870	1200	1600
Selected Images	206				523				1743			
New Extractions ↑	35	101	106	103	31	98	**134**	132	22	74	81	91
FID ↓	48.28	62.25	62.14	66.1	44.06	48.72	48.45	52.79	36.41	36.43	38.17	**31.07**

diverse generations. However, it usually comes with the price of reduced generation quality, which is quite obvious when setting τ_f above 0.35.

The filtering threshold τ_f. The filtering threshold τ_f mainly affects the number of selected images. As shown in Table 4, a smaller threshold always leads to more selected images. By setting a smaller threshold, we can collect not only more successful generations, but also more generations we don't want. In fact, the ratio of the successful generations among all the selected images will fall, as seen in the Appendix F. This could result in the approximated distribution shifting away from the true distribution of the training set and fewer extractions. This phenomenon is obvious when we reduce τ_f from 0.4 to 0.35. In all, since our prior objective is to extract training images efficiently, we deem the best-performing set of hyperparameters is $\tau_f = 0.4, c = 448$.

5.5 Qualitative Results

In this section, we show that the efficiency gain of our method doesn't mean that the successful generations are of low quality. Some of the successful generations and the corresponding training images are shown in Fig 3. Despite different backgrounds and lighting conditions, the face of the upper row is almost identical to the corresponding one below. On the one hand, this proves that our definition of extraction is adequate. On the other hand, we observe that there seems not to be any problem with the generation quality of the upper row, which shows that our method doesn't hurt the generation quality of these successful generations.



Figure 3: The extraction results on LDMs trained on CelebA-HQ and FFHQ. The images in the upper row are generated by DMs and the ones in the bottom are the matched images from the training set.

6 Conclusion

In this paper, we study the data privacy risk of unconditional DMs trained on human face datasets. We first define extraction in the context of human face images, and, based on that, we formulate the new problem of efficient extraction attacks. We firstly identify the two factors affecting the efficiency of extraction attacks: the speed of generation and the per-sample hit probability $\bar{p}$. Surprisingly, we find that SDE-DPM-Solver Lu et al. (2025), which is commonly believed to be less efficient than ODE counterpart in sampling, is more efficient here due to a larger $\bar{p}$. To further improve $\bar{p}$, we propose the Generation-Filtering-Resampling framework that adapts the distribution of initial noise to favor sampling in high-probability regions. Our experiments demonstrate that our method is far more efficient than existing ones with different kinds of DMs trained on various datasets.

REFERENCES

Tony Bonnaire, Raphaël Urfin, Giulio Biroli, Marc Mezard, et al. Why diffusion models don't memorize: The role of implicit dynamical regularization in training. In *Proceedings of Neurips 2025*. (seleziona...), 2025.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX security symposium (USENIX Security 23)*, pp. 5253–5270, 2023.

Yunhao Chen, Xingjun Ma, Difan Zou, and Yu-Gang Jiang. Extracting training data from unconditional diffusion models. *arXiv preprint arXiv:2406.12752*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *International Conference on Machine Learning*, pp. 8717–8730. PMLR, 2023.

Jan Dubiński, Antoni Kowalczuk, Stanisław Pawlak, Przemyslaw Rokita, Tomasz Trzciński, and Paweł Morawiecki. Towards more realistic membership inference attacks on large diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4860–4869, 2024.

Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David Fleet. Video diffusion models. *arXiv (Cornell University)*, abs/2204.03458, 2022. doi: 10.48550/arxiv.2204.03458.

Xin Huang, Ruizhi Shao, Qi Zhang, Hongwen Zhang, Ying Feng, Yebin Liu, and Qing Wang. Humannorm: Learning normal diffusion model for high-quality and realistic 3d human generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4568–4577, 2024.

Hyeonho Jeong, Chun-Hao Huang, Jong Ye, Niloy Mitra, and Duygu Ceylan. Track4gen: Teaching video diffusion models to track points improves video generation. *CVPR*, pp. 7276–7287, 2025. doi: 10.1109/cvpr52734.2025.00682.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Hk99zCeAb`.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36:76525–76546, 2023.

Tomoya Matsumoto, Takayuki Miura, and Naoto Yanai. Membership inference attacks against diffusion models. In *2023 IEEE Security and Privacy Workshops (SPW)*, pp. 77–83. IEEE, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 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.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

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.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6048–6058, 2023a.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36:47783–47803, 2023b.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, Yuwei Guo, Tianxing Wu, Chenyang Si, Yuming Jiang, Cunjian Chen, Chen Loy, Bo Dai, Dahua Lin, Yu Qiao, and Ziwei Liu. Lavie: High-quality video generation with cascaded latent diffusion models. *Int. J. Comput. Vis.*, 133(5):3059–3078, 2024. doi: 10.1007/s11263-024-02295-1.

Xiaoyu Wu, Jiaru Zhang, and Steven Wu. Leveraging model guidance to extract training data from personalized diffusion models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68060–68077. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wu25as.html.

Shuchen Xue, Zhaoqiang Liu, Fei Chen, Shifeng Zhang, Tianyang Hu, Enze Xie, and Zhenguo Li. Accelerating diffusion sampling with optimized time steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8292–8301, 2024.

Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6796–6807, June 2024.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Shengfang Zhai, Huanran Chen, Yinpeng Dong, Jiajun Li, Qingni Shen, Yansong Gao, Hang Su, and Yang Liu. Membership inference on text-to-image diffusion models via conditional likelihood discrepancy. *Advances in Neural Information Processing Systems*, 37:74122–74146, 2024.

Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 253–261, 2020.

Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7777–7786, 2024.

A IMPLEMENTATION DETAILS

The total number of generations For both our method and the comparing ones, we generate a total of 200,000 images in all settings. In our case, we generate 100,000 images for the Generation phase and generate the remaining 100,000 images for the Resampling phase ($k_1 = k_2 = 100,000$). The efficiency of the Resampling stage is better than the Generation stage, so allocating more generations for the Resampling stage can make the framework more efficient. However, we still need to generate sufficient images in the Generation stage for filtering. We find that simply setting $k_1 = k_2$ works well, and it already proves the effectiveness of our framework. So we believe the further tuning of $\frac{k_1}{K_2}$ is less important.

Compute All experiments are conducted on 4 NVIDIA V100 32GB SXM2.

Other hyperparameters See Tab. 5

Table 5: Hyperparameters for each setting.

Hyperparameter	LDM on FFHQ	LDM on CelebA-HQ	DDPM on CelebA-HQ
Clique size m	5	5	5
Filtering threshold τ_f	0.45	0.4	0.35
GMM components c	160	448	80
Resolution	256×256	256×256	256×256

B EXPERIMENTS ON A NON-FACE DOMAIN

We conduct experiments on a non-face domain and verify the advantage of SDE over ODE and the effectiveness of our method in Tab.6. Specifically, we test our method in LSUN-Churches Yu et al. (2015), a diverse and deduplicated scene dataset that includes outdoor scenes in different regions, seasons, weather and timing. We define extraction by SSCD similarity, and we use 0.65 as the threshold here. The results are on 100k distinct generations.

Table 6: Experiment results on LSUN-Churches Yu et al. (2015).

Method	Extraction ↑	Time ↓	Extraction / h ↑
60-step DPM-Solver++	50	5.67h	8.82
60-step SDE-DPM-Solver++	69	5.7h	12.11
GFR	**195**	**4.97h**	**39.24**

Firstly, the SDE solver achieves 12.11 extraction per hour, which is 37.3% more efficient than the ODE counterpart, which can be attributed to its larger hit rate. This shows the advantage of the SDE over ODE in extracting non-face images. Secondly, compared to the SDE solver, the GFR increases the efficiency from 12.11/h to 39.24/h, achieving a $3.2\times$ gain. This verifies the effectiveness of our method in a non-face domain.

C EXPERIMENTS ON DIFFERENT τ_e

In Sec. 3.1, we define the extraction by the facial feature similarity between two images being greater than 0.65. We show that changing the extraction threshold τ_e doesn't affect our conclusion. In other words, our method is still more efficient than SDE-DPM-Solver and DPM-Solver with a different τ_e. We evaluate the number of extractions per hour on the 1000-step DDPM sampler, the 60-step DPM-Solver, the 60-step SDE-DPM-Solver, and our method. The experiments are conducted on the LDMs Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018) with 200,000 generations.

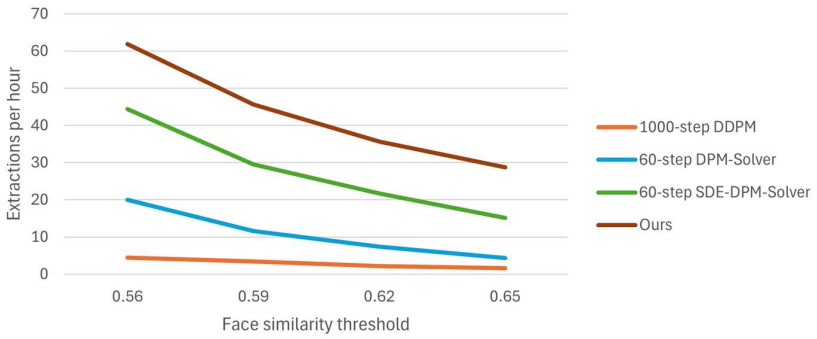


Figure 4: The number of extractions per hour with different extraction thresholds τ_e. We test the 1000-step DDPM sampler, the 60-step DPM-Solver, the 60-step SDE-DPM-Solver, and our method on an LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018).

As shown in Fig. 4, setting a larger τ_e makes fewer extractions per hour. However, our method is still more efficient than the 60-step SDE-DPM-Solver, showing that setting different τ_e doesn't affect our improvement in efficiency.

D COMPONENT ATTRIBUTION

The efficiency analysis in Sec.5.3 shows that the majority of the efficiency improvement of our GFR framework comes from the Filtering and Resampling stages. Skipping the Resampling phase would degrade GFR into a simple SDE solver, reducing extractions per hour from 28.78 to 18.06. To further validate the effectiveness of each component, we added a series of complementary controlled ablations on the CelebA-HQ LDM dataset.

Filtering stage and GMM modeling. We evaluate the individual contributions of the Filtering stage and multimodal proposal modeling in Tab 7. We fix the Generation pool to 100K images and use the same downstream Resampling protocol for all experiments. Specifically, we fit the proposal distribution using 523 images at $= 0.7T$ and generate 40K images for evaluation. We compare two image selection strategies (our Filtering stage vs. random sampling from the Generation pool) and two proposal models (a single Gaussian vs. a 448-component GMM).

Table 7: Ablation studies on the Filtering stage and GMM modeling.

Selection method	Selected	Proposal model	New extractions	FID
Random	523	448-component GMM	19	**27.77**
Filtering	523	1-component GMM	31	44.06
Filtering	523	448-component GMM	**134**	48.45

The comparison between the first and third rows shows that using filtered rather than randomly selected images increases the number of new extractions from 19 to 134, achieving a $7.1\times$ gain. This demonstrates that the Filtering stage effectively guides the learned proposal distribution toward the training data distribution. The comparison between the second and third rows further shows that replacing a single Gaussian with a multimodal GMM increases the number of new extractions from 31 to 134, achieving a $4.3\times$ gain. This demonstrates that the 448-component GMM is more effective in capturing the multimodal structure of the training data.

Filtering rule and selection metric We investigate the design choices of the Filtering stage in Tab 8. We fix the Generation stage to the same 100K images produced by a 60-step SDE-DPM-Solver and fix the Resampling stage to a 448-component GMM at $= 0.7T$, followed by 40K samples from the same 60-step SDE-DPM-Solver. Only the filtering rule or selection metric is changed. The size of cilques are chosen so that the compared methods select similar numbers of images.

Table 8: Ablation studies on the design choices of the Filtering stage.

Filtering rule	Metric	Edge threshold	Selected	New extractions	FID
Clique finding	Face similarity	0.40	523	**134**	48.45
Degree threshold	Face similarity	0.40	523	72	70.05
Clique finding	Pixel l_2	0.11	609	13	36.34
Clique finding	Pixel l_2	0.12	592	11	36.34

At the same selected-set size of 523, clique finding yields 134 new extractions, compared with 72 for degree thresholding, representing an 86% increase. This shows that mutual connectivity contributes beyond simply selecting high-degree points. Holding clique finding fixed, face similarity yields 134 new extractions, compared with only 11–13 for pixel-level l_2 distance. This supports our motivation in Sec. 3.1: pixel distance is easily dominated by background and lighting, while face similarity better captures repeated identity content.

Resampling design To isolate the contribution of resampling, we compare GFR with continuing to sample from the standard Gaussian in Tab. 9. Both settings generate 200K images in total with the same victim model and base SDE solver.

Table 9: Ablation studies on the Resampling stage.

Method / phase	Generations	Unique extractions	Extraction rate	Time
Standard 60-step SDE sampling	200K	128	15.12/h	8h 28min
GFR Generation stage	100K	78	18.06/h	4h 19min
GFR Resampling stage	100K	**158 new**	43.89 new/h	3h 36min
GFR overhead	-	-	-	17min
GFR combined	200K	**236**	**28.78/h**	**8h 12min**

Thus, replacing the second 100K standard-Gaussian samples with GMM resampling increases the final distinct extraction count from 128 to 236 while also slightly reducing end-to-end time, because denoising starts at $0.7T$. The phase times in Table 3 sum to 8h 12min, which is consistent with the reported extraction rate of $236/8.2 = 28.78$ extractions/h. The gain is therefore not explained by the graph alone or by uncounted preprocessing overhead.

E THE ROLE OF RANDOM NOISE

To show that random noise is the key factor affecting the extraction efficiency, we compare the DPM-Sovler with a series of SDE-DPM-Solvers with different noise magnitudes. Specifically, We parameterize the reverse SDE by scaling the diffusion term with η:

$$d\mathbf{x} = \left[f - \frac{1+\eta^2}{2} g^2 s_\theta \right] dt + \eta g \, d\bar{\mathbf{W}}_t. \tag{4}$$

Here, $\eta = 0$ gives the deterministic probability-flow ODE and $\eta = 1$ gives the standard reverse SDE. In a separate controlled sweep, we generated 40K images for each value using a 60-step, order-2 SDE-DPM-Solver on the CelebA-HQ LDM in Tab. 10. Runtime was essentially unchanged across the sweep and the number of extractions indicates the extraction efficiency.

The extraction count rises from 10 at $\eta = 0$ to 28 at $\eta = 0.75$ and 26 at $\eta = 1$. Thus, the effect is a tunable consequence of substantial stochasticity rather than a binary solver label. The non-monotonic optimum also shows that simply adding more noise is not always better.

F THE INFLUENCE OF τ_f

In this section, we conduct experiments on the LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018) to show the influence of the filtering threshold τ_f on the accuracy of the

Table 10: The influence of noise magnitude to the extraction efficiency.

η	Unique extractions	FID
0.00	10	23.46
0.25	12	23.33
0.50	12	22.69
0.75	**28**	**21.03**
1.00	26	23.28

clique-finding method. Specifically, we first use the 1000-step DDPM sampler to generate 100,000 images, and then we compare each of the generated images x_i with the training set to find all successful generations (there are 150 successful generations), which we denote as $S = \{x_i \in G | \max_{y_j \in D_t} \text{sim}(f(x_i), f(y_j)) > \tau_e\}$. This serves as the ground truth for our testing. After that, we use the clique-find method with different τ_f to find those successful generations among the 100,000 images. For the images $\hat{G} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_{n_f}\}$ selected by the clique-finding method with τ_f, we evaluate two metrics:

$$Recall = \frac{\sum_{x_i \in S} \mathbb{1}\{x_i \in \hat{G}\}}{|S|} \tag{5}$$

$$Precision = \frac{\sum_{\hat{x}_i \in \hat{G}} \mathbb{1}\{\hat{x}_i \in S\}}{|\hat{G}|} \tag{6}$$

where $\mathbb{1}\{\}$ is the indicator function. $Recall$ measures how many successful generations are found out of all the successful generations, and $Precision$ measures the percentage of real successful generations in all the selected images. We evaluate $Recall$ and $Precision$ under different threshold τ_f in Fig. 5. It is quite obvious that setting the threshold smaller will collect more successful generations, so the $Recall$ gets larger. However, it also brings even more images that don't belong to S, which makes $Precision$ drop. There is a trade-off between $Recall$ and $Precision$. If we set τ_f too high, even if all the selected images are successful generations, we miss the majority of the successful generations, and the learned distribution is biased. If we set τ_f too small, the successful generations only take a small fraction of the selected images, and the distribution learned from these images may shift away from the actual distribution of the training set.

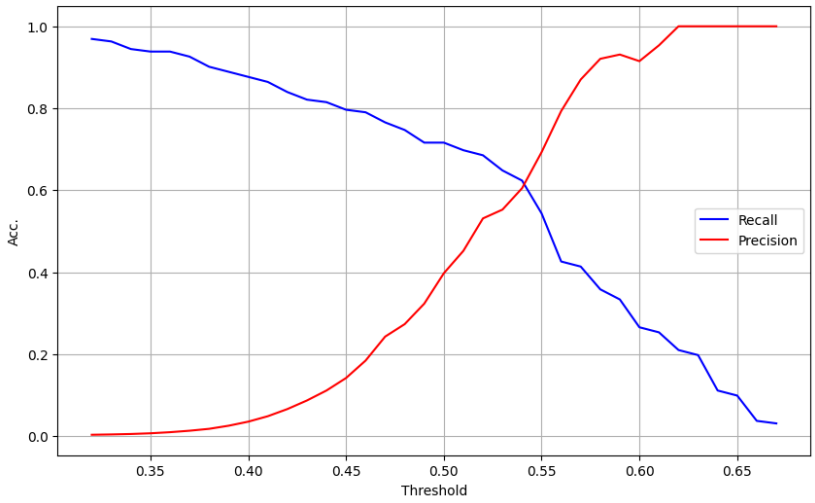


Figure 5: The $Recall$ and $Precision$ of the clique-finding method under different threshold τ_f.

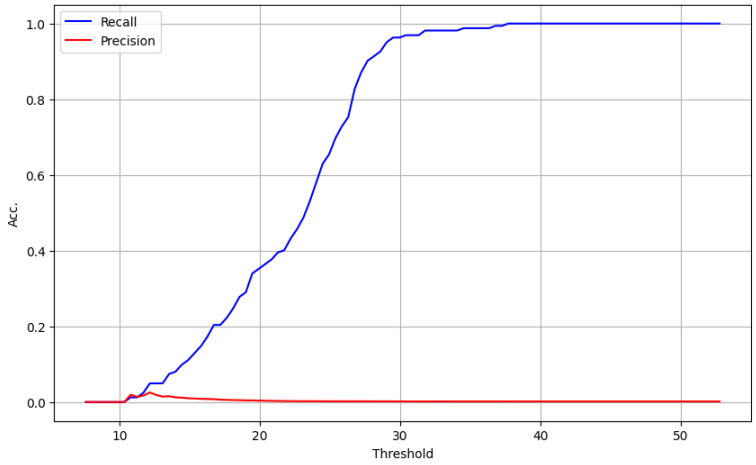


Figure 6: The $Recall$ and $Precision$ under different threshold using SecMI Duan et al. (2023).

G Can MIA methods be used in the Filtering stage?

Since the goal of the Filtering stage is to find replicas of the training images among all the generated images, it is natural to apply MIA methods here. So we use a typical MIA method for DMs, SecMI Duan et al. (2023), to replace the clique-finding method and redo the experiments in Sec. F. As shown in Fig. 6, no matter what threshold we use, $Precision$ is well below 10%. Contrary to what people generally believe, this implies that MIA can not be used for finding the replicas of the training images. The reason, we believe, is that DMs tend to reproduce the memorized images in a different way, as shown in Fig. 3. When the background or the lighting of the generated image is different from that of the corresponding training image, MIA methods are likely to fail. This reveals the limitation of the MIA methods for DMs.

H Examples of extractions on DDPM

In this section, we show some of the examples extracted from the DDPM Ho et al. (2020) pretrained on CelebA-HQ Karras et al. (2018). As shown in Fig. 7, the generated faces are almost identical to the corresponding ones below. This proves that our method can also be applied to pixel-space DMs. However, there is still a tiny difference between the faces in the first column. We believe this is caused by the difficulty of extraction attacks in pixel-space DMs. Previous study Gu et al. (2023) shows that the DMs trained on a larger resolution are less likely to memorize the training images. In our case, the pixel-space DDPM Ho et al. (2020) is trained on images with the size 256×256, which are larger than those latent (64×64) used to train the LDM Rombach et al. (2022). So it might be hard for the pixel-space DM to reproduce the exact details of the faces in the training set.

I Comparison to 1000-step DDPM sampler

In this section, we provide a qualitative comparison between our method and the 1000-step DDPM sampler Ho et al. (2020) to show that our framework doesn't impair the quality of the successful generations. We compare two methods on the LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018). For both methods, we show the 10 best-matching image pairs, each consisting of a generated image and the corresponding training image.

As shown in Fig. 8, these faces generated by our method and the 1000-step DDPM sampler are almost identical to the corresponding ones in the training set. Comparing the images generated by the two methods, there doesn't seem to be any quality problem with the images generated by our method. This proves that our framework doesn't hurt the generation quality of the successful generations.

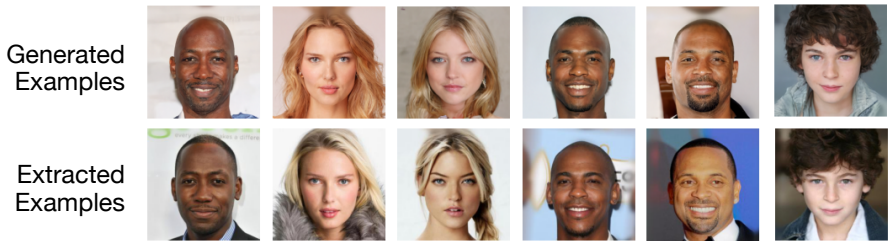


Figure 7: Some examples of the extracted images in DDPM Ho et al. (2020) training on CelebA-HQ Karras et al. (2018). The upper row of images is generated by our method and the matching training images are shown below.

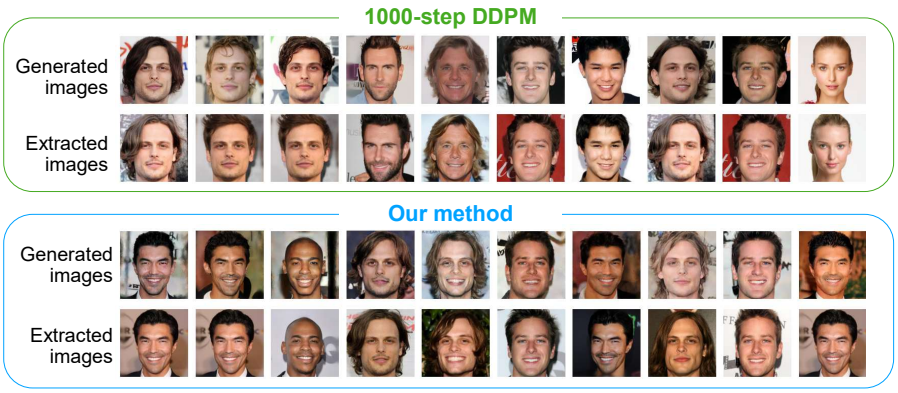


Figure 8: Qualitative comparison between our method and 1000-step DDPM sampler Ho et al. (2020). These are the most-matching examples extracted by the two methods from an LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018). For each method, the generated images are shown on the top and the matching training images are shown at the bottom.

J EFFICIENCY ANALYSIS

Results on other settings. In Sec. 5.3, we have provided a efficiency analysis on the LDM Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018). We show more results of other settings in Tab. 12 and 13.

The problem of degraded generation quality. We provide additional FID metrics in Tab. 11 12 13 for reference. We observe that the FID of the Resampling stage is larger than the baseline in all settings. We attribute the degraded generation quality to the discrepancy between our learned distribution $\hat{q}$ and the original distribution q_o of all the latents at 0.7T timestep. The original distribution q_o ensures that if the latent at 0.7T timestep falls into it, then the generated image will be of good quality. However, since the learned distribution $\hat{q}$ is modeled by a mixture of a finite number of Gaussian, there are points which $\hat{q}$ assigns non-negligible probabilities lying out of the distribution q_o. Consequently, this points generally lead to low-quality generations, which increased the overall FID metric. However, these points typically don't lead to any extraction, so this generation quality problem doesn't affect the quality of the successful generations (as shown in Fig. 8 3). Since extraction is our priority, the FID problem can be tolerated. Note that we don't calculate the FID of the combined set of stage 1 and stage 3 which is expected to lie between the FIDs of the first stage and the third stage. We believe it is not important here.

Table 11: Efficiency analysis conducted on LDM Rombach et al. (2022) trained on CelebA-HQKarras et al. (2018). The number of generations is listed in the third column. The number of (new) extractions and the number of (new) extractions per hour are listed in the fourth and fifth columns, respectively. For the third stage, we only count the number of new extractions (per hour).

Phase	Operation	Gen	Extr. ↑	Extr./h ↑	Time Cost ↓	FID ↓
Baseline	60-step Solver	200k	128	15.12	8h 28min	22.98
1-st	60-step Solver	100k	78	18.06	4h 19min	22.96
2-nd	Encoding		-		10min	-
	Clique Finding		-		1min	-
3-rd	GMM Training		-		6min	-
	42-step Solver	100k	158	43.89	3h 36min	46.34
Combined		200k	**236**	**28.78**	**8h 12min**	

Table 12: Efficiency analysis conducted on LDM Rombach et al. (2022) trained on FFHQKarras et al. (2019). The number of generations is listed in the third column. The number of (new) extractions and the number of (new) extractions per hour are listed in the fourth and fifth columns, respectively. For the baseline and the first stage, we show the number of extractions (per hour). For the third stage, we only count the number of new extractions (per hour) that have not been counted in the first stage.

Phase	Operation	Gen	Extr.	Extr./h	Time Cost	FID
Baseline	60-step Solver	200k	82	9.73	8h 26min	23
1-st	60-step Solver	100k	49	10.94	4h 29min	22.98
2-nd	Encoding		-		10min	-
	Clique Finding		-		1min	-
3-rd	GMM Training		-		2min	-
	42-step Solver	100k	138	40.83	3h 23min	30.92
Combined		200k	**187**	**23.14**	**8h 5min**	

K DUPLICATED EXTRACTIONS IN RESAMPLING STAGE

In our GFR framework, the distribution we modeled in the Resampling stage is learned from the generated images in the Generation stage. So it is natural to ask: "Does the Resampling stage extract

Table 13: Efficiency analysis conducted on DDPM Ho et al. (2020) trained on CelebA-HQKarras et al. (2018). The number of generations is listed in the third column. The number of (new) extractions and the number of (new) extractions per hour are listed in the fourth and fifth columns, respectively. For the baseline and the first stage, we show the number of extractions (per hour). For the third stage, we only count the number of new extractions (per hour) that have not been counted in the first stage.

Phase	Operation	Gen	Extr.	Extr./h	Time Cost	FID
Baseline	60-step Solver++	200k	16	0.81	19h 50min	22.57
1-st	60-step Solver++	100k	9	0.83	10h 53min	22.4
2-nd	Encoding		-		10min	-
	Clique Finding		-		1min	-
3-rd	GMM training		-		5min	-
	42-step Solver++	100k	43	5.3	8h 8min	44.21
Combined		200k	**52**	**2.7**	**19h 17min**	

the same images as the Generation stage?" In this section, we show that this duplication phenomenon exists, but it is not serious. We evaluate the number of extractions and new extractions (the ones that are not included in the Generation stage) on the Resampling stage in the same experiments introduced in Sec. 5.2. If the new extractions only takes up a small fraction of the whole set, the duplication behavior is serious. This phenomenon will indicate that the distribution learned in the Resampling stage overfits on the images generated in the Generation stage, and it is more likely to extract the repeated training examples.

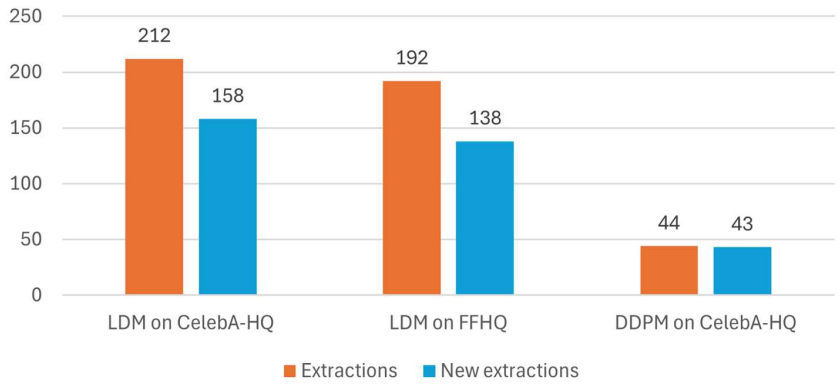


Figure 9: The number of extractions and **new** extractions for the Resample stage of our method. We test on 2 LDMs Rombach et al. (2022) trained on CelebA-HQ Karras et al. (2018) and FFHQ Karras et al. (2019), respectively, and a DDPM Ho et al. (2020) trained on CelebA-HQ Karras et al. (2018). The orange bars are the number of extractions for each setting, and blue bars are the number of **new** extractions.

As shown in Fig. 9, the new generations take up to 70% of the total extractions, which shows that the repeated extractions only account for a small part. Surprisingly, in the experiments on the DDPM Ho et al. (2020) model, there is only 1 repeated extraction out of the 44 extractions. This proves that the learned distribution favors new extraction.

L Can our method be applied to other diffusion models?

Conditional diffusion models For conditional DMs Peebles & Xie (2023) that use the class label as their conditions, it is technically feasible for our method to be applied to improve the extraction efficiency. However, for a typical image classification dataset such as ImageNet Deng et al. (2009), extraction is not well defined. We argue that defining extraction with l_2 distance in pixel space is

inappropriate. In our experiment, we use a 250-step DDPM scheduler Ho et al. (2020) to generate 100, 000 images with an ADM [2] Dhariwal & Nichol (2021) trained on ImageNet Deng et al. (2009). Then we evaluate the extraction result on them. As shown in Fig. 10, the extracted images are mostly of a small object in a simple background. Under this definition of extraction, these kind of images are especially easier to extract because any generated image with the same background can match them. However, the objects in the generated image and the matching training image are usually different, which is in fact meaningless.



Figure 10: Some of the extracted examples from ImageNet Deng et al. (2009). The extracted training images are shown in the bottom row, and the corresponding generated images are in the first row. With the ADM Dhariwal & Nichol (2021) as the victim model, we use the 250-step DDPM scheduler Ho et al. (2020) for sampling. The threshold for the l_2 distance is set to 0.1 following Carlini et al. (2023).

Rectified Flow Rectified flow Liu et al. (2023) aims at learning the ODE to follow a straight path between points of 2 distinct distributions, which was originally proposed to accelerate the sampling of deterministic ODE solvers. Though the extraction efficiency problem of this method is less serious due to its high sampling speed, our method can still be applied to them. However, since stochasticity in the generation process is the key to more extractions, one problem to solve is to find an efficient indeterministic sampler for Rectified flow. We leave this problem to future works.

M LIMITATIONS

In this paper, we focus on the efficient extraction attacks on DMs trained on human face datasets. We don't extend our discussion to other datasets for two reasons: (1) The privacy of human faces is what concerns people most. (2) It can be hard to define extraction in other datasets. As mentioned in Sec. 3.1, DMs tend to reproduce memorized contents in different backgrounds or lighting. This makes the typical l_2 distance in pixel space less accurate. It remains challenging to define extraction for other datasets like ImageNet Deng et al. (2009) and Laion Schuhmann et al. (2021).

The clique-finding method in the Filtering stage of our framework is based on the assumption that the duplicated generations are more likely to be close to the training images. This assumption, which has been proven to hold in many cases Carlini et al. (2023); Wu et al. (2025), is vital for our method.

N BROADER IMPACTS

This paper reveals a critical and previously underexplored privacy vulnerability in unconditional diffusion models trained on human face datasets. While our work significantly improves the efficiency of training data extraction attacks, it carries important dual-use implications.

On the negative side, the proposed "Generation-Filtering-Resampling" framework and the insight regarding SDE solvers could be abused by malicious actors to extract sensitive, private facial images from deployed generative models at a much faster rate. This exacerbates the risks of identity theft, unauthorized deepfake creation, and severe privacy violations, particularly for individuals whose data was scraped without explicit consent.

[2] checkpoint from https://github.com/openai/guided-diffusion/tree/main

On the positive side, uncovering this vulnerability is a necessary step toward building trustworthy and secure AI. By exposing the data privacy risk of diffusion models, we push the community to re-evaluate the privacy guarantees of generative models.