Latent Diffusion Inversion Requires Understanding the Latent Space

Mingxing Rao, Bowen Qu, Daniel Moyer
Vanderbilt University
Nashville, TN 37235,USA
{mingxing.rao, bowen.qu, daniel.moyer}@vanderbilt.edu

Abstract

The recovery of training data from generative models ("model inversion") has been extensively studied for diffusion models in the data domain. The encoder/decoder pair and corresponding latent codes have largely been ignored by inversion techniques applied to latent space generative models, e.g., Latent Diffusion models (LDMs). In this work we describe two key findings: (1) The diffusion model exhibits non-uniform memorization across latent codes, tending to overfit samples located in high-distortion regions of the decoder pullback metric. (2) Even within a single latent code, different dimensions contribute unequally to memorization. We introduce a principled method to rank latent dimensions by their per-dimensional contribution to the decoder pullback metric, identifying those most responsible for memorization. Empirically, removing less-memorizing dimensions when computing attack statistics for score-based membership inference attacker significantly improves performance, with average AUROC gains of 2.7% and substantial increases in TPR@1%FPR (6.42%) across diverse datasets including CIFAR-10, CelebA, ImageNet-1K, Pokémon, MS-COCO, and Flickr. This indicates stronger confidence in identifying members under extremely low false-positive tolerance. Our results highlight the overlooked influence of the auto-encoder geometry on LDM memorization and provide a new perspective for analyzing privacy risks in diffusion-based generative models.

1. Introduction

Model inversion is the task of recovering training samples from a trained generative model. The generative models ostensibly should be fitting an underlying data distribution, so finding the individual training points implies an overfitting, and moreover may be informative of pathological structures in fitted models. The inversion of diffusion models has been extensively explored in prior work [4, 7, 10, 26, 27, 32]. However, for Latent Diffusion Models (LDMs) [22], existing research at onces (1) predominantly focuses on invert-

ing the diffusion process itself, treating the latent space as a fixed substrate and largely overlooking the role of the associated variational auto-encoder (VAE), and (2) has significantly reduced inversion performance [21] relative to data-domain diffusion processes.

A crucial precursor to model inversion is the *membership inference attack* (MIA), where attack methods attempt to distinguish training examples (members) from non-training examples (non-members). This measures how well we can identify training-points without considering the relatively unrelated search problem over the space of images. Many effective MIA methods have been proposed for diffusion models [4, 4, 9, 13, 19, 30]. A model with stronger training-set memorization (overfit) typically shows higher MIA vulnerability; thus, the sensitivity to MIA serves as a practical metric for evaluating memorization.

Prior work [21] shows that varying the latent-space regularization leads to substantially different degrees of vulnerability to membership inference, indicating that the structure of the latent space influences model memorization. In this paper we take this a step further, demonstrating that geometric properties of the decoder implicate upon membership inference performance. Based on statistics of the decoder geometry, we construct a filtering procedure that improves membership inference performance *across all tested methods*; this demonstrates that these local statistics are connected with overfit phenomena, and not simply biases or idiosyncrasies in particular model inversion techniques.

To characterize geometric properties of the decoder, we employ the *pullback metric* from Riemannian geometry [2, 3, 5]. This is metric tensor associated with the decoder mapping (the "pulling back" of the data-space metric to the latent space), and its determinant measures the local volume change induced by the decoder [2], which we call the distortion. Empirically, we find that latent diffusion models tend to memorize samples located in regions of *high* decoder distortion more strongly than those in low-distortion regions (diagrammed in Fig. 1). In contrast, if the latent space exhibits nearly uniform distortion, memorization becomes correspondingly uniform. The strong cor-
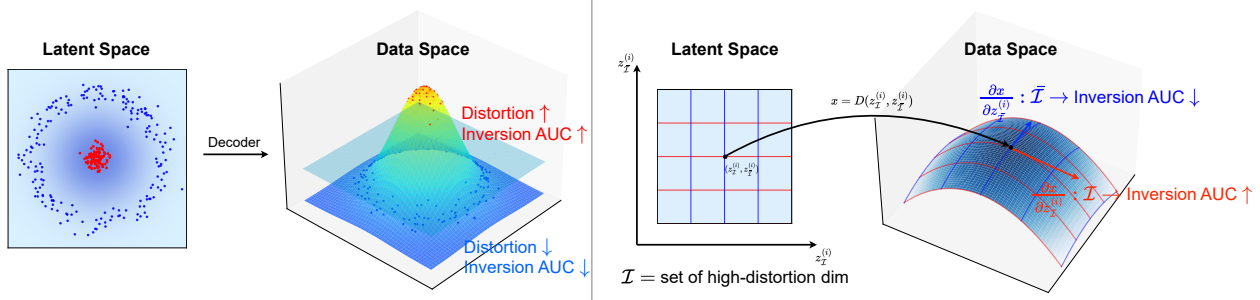
Figure 1. **Left:** Points mapped to high-distortion regions of the latent space (red) are more vulnerable to inversion and exhibit stronger memorization compared to those in low-distortion regions (blue). **Right:** Latent grid lines and their decoded counterparts show that high-distortion latent dimensions (red) induce larger changes in data space, whereas low-distortion dimensions (blue) induce smaller changes. This reflects that latent diffusion models memorize high-distortion dimensions more strongly than low-distortion ones.

relation between decoder's local distortion and memorization indicates that the choice of encoder/decoder architecture plays an important role in score field learning. Experiments to support these claims are presented in Section 4.2.

Previous work observed that data domain diffusion denoisers preferentially reconstruct high-variance directions while oversmoothing low-variance directions [12, 18, 28]. These directions correspond to eigenvectors of the data covariance matrix. Although this analogy is not rigorously transported to the latent space without significant computation, whose nonlinear encoding behavior can differ substantially point-to-point, it suggests that certain latent directions may contribute disproportionately to memorization. We improve the inversion performance by preferentially including or excluding different latent space dimensions based on the *per-dimensional contribution* to the local distortion induced by the decoder. In Section 3.3, we introduce a computationally efficient method to quantify these contributions, diagrammed in Fig. 1 (right). We empirically validate this idea by selectively removing the less-contributing (alternatively, less-memorizing) dimensions and evaluating the resulting membership inference attacks. Across multiple datasets, this dimensional masking consistently improves attack performance: AUROC increases by 1–4%, and TPR@1%FPR improves by 1–15%.

In summary, our contributions are the following:

1. We show that latent diffusion models exhibit *spatially non-uniform memorization* in latent space and demonstrate that this behavior is tightly correlated with the *local distortion* measured by the decoder pullback metric.
2. We propose a principled, geometry-driven measure of *per-dimensional memorization strength* based on each latent dimension's contribution to decoder distortion.
3. We provide extensive empirical evidence showing that removing less-memorizing latent dimensions of attack vector before computing the norm statistics consistently improves all metrics over all MIA methods.

All data splits, model checkpoints, training/fine-tuning scripts, and testing code are released on our GitHub repository. [withheld for review]

2. Related Work

Riemannian Geometry of Variational Autoencoders. A line of work studies the Riemannian geometry induced by deep generative models, and VAEs in particular, via the pullback metric of the decoder. Arvanitidis et al. show that the generator of a VAE induces a stochastic Riemannian metric on the latent space and that geodesics and distances computed under this metric lead to more semantically meaningful interpolations, improved sampling, and better clustering [2, 3]. Shao et al. [24] further develop algorithms for computing geodesics and parallel transport on the manifolds defined by deep generative models. Subsequent work has used this geometric viewpoint to regularize latent spaces, design geometry-aware VAEs, and perform data augmentation in small-sample regimes by exploiting the learned Riemannian structure [3, 5]. Different from these works, which primarily use the pullback metric to study interpolation, representation quality, and manifold geometry, our goal is different: we treat the decoder-induced pullback metric as a *privacy-relevant signal*. We estimate local distortion for high-dimensional VAEs used in LDMs, and we show that this distortion reveals where the latent diffusion model memorizes most strongly and which latent dimensions contribute most to membership leakage.

Latent-domain diffusion models. LDMs [22] factor generative modeling into a VAE and a diffusion model trained in the resulting latent space, dramatically reducing computational cost while preserving visual fidelity. Subsequent work has largely followed this paradigm but focused on improving synthesis quality, efficiency, or conditioning, for example by replacing the standard VAE with alternative autoencoder families or representation autoen-

coders [25, 31], or by scaling latent diffusion to new modalities and tasks. In parallel, a growing literature analyzes privacy and memorization in diffusion models, proposing increasingly strong membership inference and data extraction attacks [4, 8, 9, 13, 20, 30], and several works study frequency-domain behavior and "pass-through" phenomena of data-domain denoisers [12, 14, 15, 18, 28]. However, these analyses either operate in pixel space or treat the autoencoder in LDMs as a fixed preprocessor. By contrast, we explicitly attribute memorization behavior of latent-space diffusion models to the *geometry* of the encoder–decoder pair. Our results show that decoder-induced local distortion in the VAE latent space controls spatially non-uniform memorization and that per-dimension contributions to this distortion identify latent directions that dominate membership leakage, bringing attention to the encoder–decoder architecture as a primary factor in the inversion properties of LDMs.

Frequency analysis on data-domain diffusion models. Prior work [18, 28] has shown that pixel-domain diffusion model denoisers often struggle with high-frequency components, which correspond to low-variance directions in the eigen-space of the data covariance. In particular, these denoisers tend to suppress high-frequency content and implicitly project reconstructions toward high-variance, low-frequency directions. However, in latent-domain diffusion models, the dynamics are fundamentally different: the diffusion process operates on nonlinear, semantically structured latent representations produced by an encoder, and the mapping back to the image domain is mediated by a decoder with nontrivial geometry. We empirically show that the local distortion of an image in latent space is not fully explained by its high-frequency energy in the pixel domain (see Table 3). This suggests that frequency-based analyses developed for data-domain diffusion models [14, 15] may not directly transfer to latent-domain models.

Moreover, Lian et al. [15] report that suppressing high-frequency components in the denoised images can enhance the separation between membership attack statistics on training versus non-training data. In our experiments with Stable Diffusion, we do *not* observe a similar improvement when applying analogous high-frequency suppression (see Appendix, Section E). Taken together, these findings indicate that latent-domain diffusion models require a different analytical perspective: rather than relying solely on frequency-based reasoning in pixel space, one must account for the nonlinear encoder–decoder geometry and its induced distortion in latent space.

3. Methodology

Terminology. Unless otherwise specified, the term *latent space* refers to the latent representation learned by the VAE
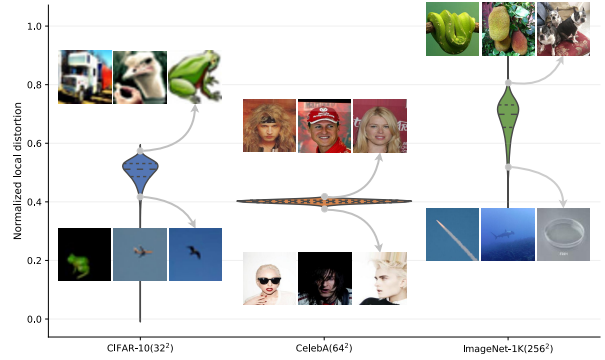

Figure 2. Each violin plot shows the distribution of the decoder-induced local distortion, with representative samples displayed at high-distortion (top) and low-distortion (bottom) regions. CelebA shows an almost uniform distortion distribution, while CIFAR-10 and ImageNet-1K display larger variation across samples. Refer to the appendix for additional samples.

component of LDM. The diffusion process operating in this latent representation is referred to as the *latent space diffusion model*.

We outline the conceptual development of our approach here. Section 3.1 summarizes the state-of-the-art score-based membership inference attacker used to evaluate model memorization. Section 3.2 introduces the definition of *local distortion*, a key geometric property of the VAE decoder that quantifies how infinitesimal regions in latent space are stretched in the data space. In Fig. 3 of Section 4.2, we demonstrate that samples located in regions of higher local distortion are more susceptible to membership attacks, revealing a non-uniform latent space memorization of the latent space diffusion model.

Motivated by this, we hypothesize that beyond non-uniform memorization across data samples, the latent space diffusion model exhibits *dimension-wise non-uniform memorization*—that is, high-distorted dimensions tend to leak more membership information. Section 3.3 introduces a simple method to quantify the influence of each dimension on memorization, and Section 4 presents empirical evidence supporting this hypothesis.

3.1. Score-based Membership Inference Attacks

Membership Inference Attacks (MIA) are pre-cursor tasks to model inversion. They measure the separability (conditional on an attack methodology/model) of seen training and unseen test data given a trained diffusion model and a threat model. While not performing the full search required by model inversion, MIA demonstrates that candidate data-points can be classified as *in training* or *out of training*. For our threat model, we consider a common grey-box [4, 8, 9, 13, 19, 30] scenario, where the attack can ma-
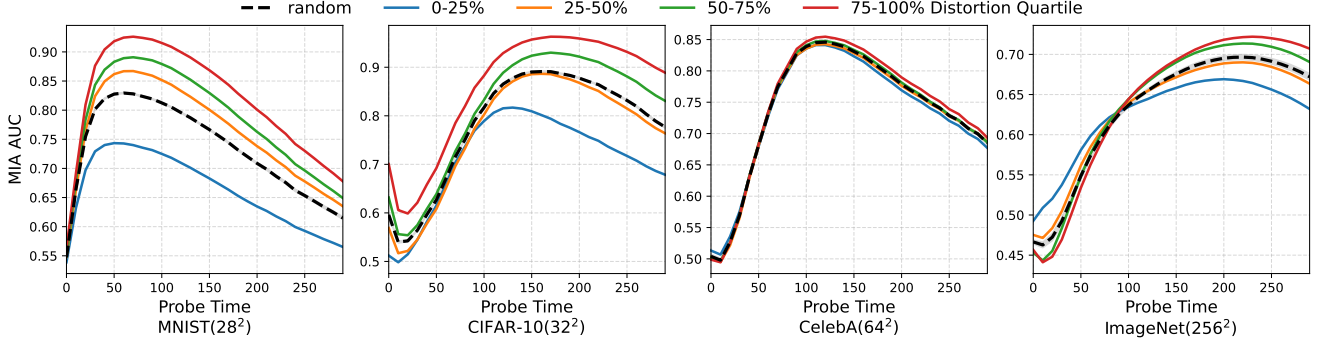
Figure 3. Membership inference AUC (measured by SimA) at different times for four datasets, stratified by quartiles of local distortion of decoder (0–25%, 25–50%, 50–75%, 75–100%). Quartile thresholds are computed jointly over members and held-out samples. Attacks are evaluated separately within each quartile, and a random baseline (mean and variance over ten trials) is shown for comparison. Higher-distortion quartiles consistently yield higher attack AUC.

nipulate model inputs but should be agnostic to particular architectures or weight configurations.

The success of these attacks can be viewed as a proxy measurement for overfitting and memorization of the training set. While not exactly equivalent, at a high level their success indicates that the generative model being attacked has information about the training dataset beyond its underlying generative distribution, i.e., about the specific training dataset. Many of the attacks use this overfitting intuition to construct their methods, using overfit behaviors to identify training data.

A baseline "loss" attack (also referred to "naive" method) evaluates the denoising objective on the queried point by sampling $\varepsilon \sim \mathcal{N}(0, I)$, forming $x_t^\star = \sqrt{\bar{\alpha}_t}\, x^\star + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$, and scoring

$$\text{LOSS}(x^\star, t) \;=\; \big\| \varepsilon - \hat{\varepsilon}_\theta(x_t^\star, t) \big\|,$$

often averaged over several ε draws [19]. SecMI further exploits diffusion structure by measuring a single–step *posterior estimation error* ("t-error") at a chosen timestep t using deterministic DDIM mappings. Let Φ_θ and Ψ_θ denote the deterministic reverse/denoise operators; with $\tilde{x}_t = \Phi_\theta(x^\star, t)$, SecMI's statistic is

$$\text{SecMI}(x^\star, t) := \big\| \Psi_\theta\big(\Phi_\theta(\tilde{x}_t, t), t\big) - \tilde{x}_t \big\|_2,$$

which is thresholded (or fed to a small attack network) to decide membership [8]. PIA attains similar accuracy with far fewer queries by using the model's own $t{=}0$ denoiser output as a *proximal initialization* and then computing a forward/backward consistency error built from this initialization [13]. For $\tilde{x}_t^\star = \sqrt{\bar{\alpha}_t}x^* + \sqrt{1 - \bar{\alpha}_t}\hat{\varepsilon}_\theta(x, t = 0)$,

$$\text{PIA}(x^\star, t) = \big\| \hat{\varepsilon}_\theta(x^\star, t = 0) - \hat{\varepsilon}_\theta(\tilde{x}_t^\star, t) \big\| \quad (1)$$

Finally, SimA [21] is the scaled Loss criteria at $\varepsilon = 0$ (point estimate), which is the mean and mode of ε distribution.

This is inherently a direct score estimation.

$$\text{SIMA}(x^\star, t) \;=\; \big\| \hat{\varepsilon}_\theta(x^\star, t) \big\|,$$

motivated by theory linking the expected denoiser to a kernel–weighted local mean of nearby training samples. Empirically, SimA matches or exceeds multi–query baselines while being simpler and faster; hence we adopt SimA to quantify the model memoriztion in most experiments [21].

3.2. VAE Decoder Pullback Metric

In a Variational Autoencoders (VAE), for many common architectures the decoder defines a smooth mapping

$$D : \mathcal{Z} \to \mathcal{X},$$

that transforms a latent variable $z \in \mathcal{Z}$ into a data point $x = D(z) \in \mathcal{X}$. While $\mathcal{Z}$ is usually assumed to have Euclidean structure, this assumption is also usually incorrect when viewed in terms of the decoder and $\mathcal{X}$: the nonlinear decoder distorts distances, making Euclidean geometry inapplicable. Instead, the decoder induces a *Riemannian metric* on $\mathcal{Z}$, known as the *pullback metric* [2, 3, 24].

We write the Jacobian of the decoder at point z as $J_D(z) = \frac{\partial D(z)}{\partial z}$. The pullback metric is then

$$G(z) = J_D(z)^\top J_D(z), \quad (2)$$

which is a symmetric positive semi-definite tensor describing how local directions in $\mathcal{Z}$ are stretched or compressed when mapped to $\mathcal{X}$. For an infinitesimal displacement $\mathrm{d}z$, the induced distance in data space is

$$\|\mathrm{d}x\|^2 = \mathrm{d}z^\top G(z)\, \mathrm{d}z. \quad (3)$$

A scalar measure of local geometric distortion is given by the volume change factor

$$\text{Distortion}(z) = \sqrt{\det(G(z))}. \quad (4)$$

To compute this stably, we use its logarithmic form

$$\log \sqrt{\det(G(z))} = \sum_i \log \sigma_i(J_D(z)), \quad (5)$$

where $\sigma_i(J_D(z))$ denote the singular values of the decoder Jacobian $J_D(z)$. Since computing all of the singular values of the Jacobian is intractable for high-dimensional latent spaces, we approximate the local distortion via a truncated spectral estimate, employing a *Matrix-free Randomized Singular Value Decomposition (SVD)* to extract the top-K singular values of $J_D(z)$ (see the pseudo-code in Appendix A). Specifically, $S_K = \sum_{i=1}^{K} \log \sigma_i(J_D(z))$, for descending sorted σ_i.

Large singular values appear to correspond to directions that produce fine-grained, spatially detailed (high-frequency) pixel changes. Note that in VAE theory, there is little to no reason that this should be true, but our empirical evidence across datasets shows it to be a consistent trend. If one image's S_K is much larger, Fig. 2 shows this image is usually semantically meaningful, which we assume that image's local neighborhood contains more high-frequency detail — edges, texture — while the other image's neighborhood is smoother or simpler. More examples can be found in Appendix F. Nevertheless, to our knowledge, no prior work demonstrates that greater distortion implies greater semantic variation.

3.3. Quantifying Per-dimension Influence on Latent Memorization

Influence Measure Definition: As we show empirically in Fig. 3 of Section 4.2, distortion is related to overfit phenomena. We hypothesize that local to a given datapoint, latent dimensions contribute in differing amounts to these overfit phenomena. Some latent dimensions contribute more strongly to the decoder's local distortion are more susceptible to memorization. To quantify this relationship, we define a per-dimension influence measure that captures each coordinate's relative contribution to the VAE decoder's Jacobian magnitude:

$$\mathrm{Infl}_i(z) := \left\| \frac{\partial x}{\partial z_i} \right\|_2^2 = e_i^\top G(z)\, e_i = G_{ii}(z). \quad (6)$$

Masking scheme for attack statistics: Given any scalar attack statistic, a **general paradigm** is $\|S(z)\|$, where $S(z)$ is an attack vector (e.g., score-loss, noise-prediction difference or loss aggregated across timestep) in latent space. We form a *dimension-masked* statistic by keeping only a subset $\mathcal{I} \subset [d]$, where d is the number of dimensions on latent space:

$$\|S_{\mathcal{I}}(z)\| := \|S(z) \odot \mathbb{I}_{\mathcal{I}}(i)\|, \quad \mathbb{I}_{\mathcal{I}}(i) = \begin{cases} 1, & i \in \mathcal{I}, \\ 0, & i \notin \mathcal{I}. \end{cases} \quad (7)$$

We choose $\mathcal{I}$ as the top-k% coordinates ranked by Infl_i. Masking out low-influence coordinates removes directions that contribute little but add noise to attack statistics. Empirically, this method *increases* AUC and markedly boosts TPR@1%FPR across datasets. The results are on Table 2 and 2. Noticeably, the proposed method depends on VAE only, which indicates that latent diffusion inversion requires understanding the latent space of VAE.

Information allocation via the decoder geometry: The pullback metric $G(z)$ determines the local volume element $\sqrt{\det G(z)}$ and acts as a data-dependent conditioning matrix on the training signal that the latent UNet receives through the reconstruction path. Coordinates with larger G_{ii} correspond to directions where small latent changes expand more in pixel space, thus receiving *higher effective signal-to-noise* during training and being more likely to carry sample-specific information. Conversely, coordinates with tiny G_{ii} transmit weak, noisy supervision and dilute $S(z)$ when included.

Hutchinson Estimator: Instead of explicitly computing the diagonal entries of the pullback metric $G_{ii}(z) = \|J_D(z)e_i\|_2^2$, which requires constructing the full decoder Jacobian $J_D(z)$, we estimate them efficiently using the *Hutchinson stochastic trace estimator*. For any function $D : \mathcal{Z} \to \mathcal{X}$ with Jacobian $J_D(z)$, the diagonal of the Gram matrix can be expressed as

$$\mathrm{diag}(J_D^\top J_D) = \mathbb{E}_{v \sim \mathcal{N}(0,I)}\big[(J_D^\top v) \odot (J_D^\top v)\big], \quad (8)$$

where $\odot$ denotes the elementwise product. In practice, this expectation is approximated by Monte Carlo averaging over n_{mc} random vectors v. For each latent code z, we compute $J_D^\top v$ implicitly using reverse–mode autodifferentiation, avoiding explicit Jacobian construction. The per–dimension influence is then defined as

$$\mathrm{Infl}_i(z) = \tfrac{1}{2} \log\big(\mathbb{E}_v\big[(J_D^\top v)_i^2\big] + \varepsilon\big), \quad (9)$$

where ε is a small constant for numerical stability.

This estimator takes $\mathcal{O}(n_{\mathrm{mc}} T_D)$, which scales linearly in both the number of latent samples and the number of Monte Carlo probes, and independent of the explicit Jacobian size. Empirically, we found $n_{\mathrm{mc}}=8$ is good enough for image datasets of all resolutions. A pseudocode is provided in appendix D.

4. Experiments

4.1. Setup

We tested our findings on **six datasets** spanning image resolutions from 32^2 to 512^2 (see Appendix C for detailed dataset descriptions and train/validation splits). All experiments are conducted under four distinct threat models and executed on eight NVIDIA A40 GPUs.

Table 1. Attack performance of four baseline methods used to evaluate memorization of **LDMs**. The *filtered* setting removes the top 40% least-memorizing latent dimensions of the attack vector (as ranked by our method) before computing the norm. The first row provides an ablation in which 40% of dimensions are *randomly* removed. **Across all datasets and attack methods, the proposed *filtered* setting improves all metrics.**

Method	CIFAR-10 (%)			CelebA (%)			ImageNet (%)		
	AUC↑	ASR↑	TPR@1%FPR↑	AUC↑	ASR↑	TPR@1%FPR↑	AUC↑	ASR↑	TPR@1%FPR↑
SimA (random drop)	86.44	78.80	15.92	83.05	75.69	10.20	69.30	64.90	4.16
SimA	89.10	81.63	19.88	84.66	77.04	11.09	69.62	64.92	3.87
SimA (filtered)	91.26	83.58	24.56	88.18	80.25	17.03	72.55	67.01	7.77
Loss	73.28	67.48	7.86	68.65	63.47	6.39	67.49	63.16	4.09
Loss (filtered)	75.87	69.81	9.69	70.12	64.68	7.16	69.99	64.68	7.01
SecMI	87.42	80.12	19.11	83.09	75.85	10.53	68.21	63.44	3.71
SecMI (filtered)	89.93	81.97	23.84	86.74	78.92	16.47	71.02	65.88	7.49
PIA	85.73	78.19	15.90	83.36	75.87	9.41	66.09	62.52	2.95
PIA (filtered)	88.49	80.65	20.08	87.41	79.28	21.13	70.74	65.58	4.75
Mean Δ (%)	+2.70	+2.26	+4.80	+3.52	+3.38	+5.83	+3.53	+2.93	+2.74
Min Δ (%)	+2.16	+1.85	+1.83	+1.47	+1.21	+0.77	+2.50	+1.52	+1.80

Table 2. Attack performance of four baseline methods used to evaluate memorization of **Stable Diffusion**. The *filtered* setting removes the top 40% least-memorizing latent dimensions of the attack vector (as ranked by our method) before computing the norm. The first row provides an ablation in which 40% of dimensions are *randomly* removed. **Across all datasets and attack methods, the proposed *filtered* setting improves all metrics.**

Method	Pokémon (%)			MS-COCO (%)			Flickr (%)		
	AUC↑	ASR↑	TPR@1%FPR↑	AUC↑	ASR↑	TPR@1%FPR↑	AUC↑	ASR↑	TPR@1%FPR↑
SimA (random drop)	93.42	87.63	17.99	93.57	87.12	28.12	69.85	65.66	2.79
SimA	93.50	87.87	20.38	93.71	87.34	29.80	70.04	65.96	2.59
SimA (filtered)	94.83	89.68	52.04	96.86	92.10	49.44	73.61	68.45	3.49
Loss	92.03	85.47	40.77	83.26	75.80	13.48	63.13	59.74	2.35
Loss (filtered)	92.67	86.07	46.28	85.78	78.18	18.68	64.57	60.49	2.81
SecMI	88.12	81.04	29.02	91.35	84.00	39.16	71.81	66.54	5.29
SecMI (filtered)	89.37	82.47	30.94	93.77	87.46	50.40	74.16	68.51	6.46
PIA	94.93	90.52	32.85	92.51	85.72	24.76	68.88	64.61	2.67
PIA (filtered)	96.23	91.47	53.00	96.03	90.30	43.24	71.85	67.11	3.46
Mean Δ (%)	+1.87	+1.61	+12.19	+2.61	+3.43	+11.99	+2.09	+1.99	+0.96
Min Δ (%)	+0.64	+0.60	+1.92	+2.42	+2.38	+5.20	+1.44	+0.75	+0.46

Evaluation Metrics: We evaluated attack performance using several metrics: ASR (attack success rate, i.e., membership inference accuracy), AUC (Area Under ROC Curve), TPR at 1% FPR (TPR@1%FPR), and the number of queries per attack (#Query). The TPR@1%FPR is computed by selecting the threshold τ at which the false positive rate falls just below 0.01, and reporting the corresponding true positive rate at that operating point. The ASR is defined as the maximum accuracy achieved over all thresholds, i.e. $\tau^* = \max_\tau \frac{1}{2}\big(\text{TPR}(\tau) + 1 - \text{FPR}(\tau)\big)$ in the balanced setting. The AUC is computed as the trapezoidal integral of $\text{TPR}(\tau)$ against $\text{FPR}(\tau)$ across all thresholds.

Latent Diffusion Model: For CIFAR-10 and CelebA [17], we trained a β-VAE [11] from scratch on the *member* set. Then we use the latent representations of member set to train a latent space diffusion model unconditionally while VAE encoder-decoder is frozen. From each training split we subsample n images and partition them equally into a *member* set and a *held-out* set. ImageNet-1K [23] is trained at two different points. (1) We di-

rectly employ the publicly released autoencoder model `stabilityai/sd-vae-ft-mse` [1] without further fine-tuning. This model, developed by Stability AI, serves as the standard latent autoencoder used in the Stable Diffusion v1 framework. (2) The latent space diffusion model is also trained from scratch. We train it conditonally by using the architecture components from diffuser, which we guarantee have a similar architecture to vanilla LDM [22]. The KL-regularization (β) for three datasets are $1e - 2$, $1e - 3$, $1e - 6$. With higher resolution, one expects to reduce the β to ensure the same scale of regularization across datasets. and The architecture details are released in Appendix G.

Pre-trained Latent Diffusion Models: For Pokémon[6], COCO2017-Val [16], and Flickr30k [29], we fine-tuned Stable Diffusion v1-4[1] on a randomly selected subset of each training split, reserving an equally sized subset as the held-out set. None of these datasets are in the original

pre-training corpus.

Stable Diffusion v1-4 was fine-tuned for **15k**, **100k**, and **200k** steps on the member splits of Pokémon, COCO-2017-Val, and Flickr30k, respectively, with a fixed learning rate of 1×10^{-5} (AdamW). Additionally, we adopted the default data augmentation (Random-Crop and Random-Flip) while training.

4.2. Experiments on local distortion vs. memorization

To compute the local distortion at each latent point, we employ our matrix-free randomized SVD procedure (Appendix 1). This algorithm evaluates Jacobian–vector products using `jvp`/`vjp` primitives in PyTorch, thereby avoiding explicit construction of the full decoder Jacobian. When automatic differentiation is unavailable, we find that finite-difference approximations provide a sufficiently accurate substitute for `jvp`. Unless otherwise stated, we use fixed randomized SVD hyperparameters across all experiments: target rank $k = 20$, oversampling parameter $p = 30$, and $q = 2$ power iterations, which balance computational efficiency and numerical stability. For measuring memorization, we primarily adopt SimA [21] due to its simplicity and effectiveness.

Figure 3 summarizes our results. For each dataset, we perform membership inference attacks at multiple diffusion timesteps t (x-axis) and report the resulting AUC (y-axis). We partition both the member set and the held-out set into four quartiles according to their local distortion values and evaluate attacks on each quartile independently. We set the quartile threshold through a joint set of member and held-out. For example, the top (75%–100%) quartile corresponds to samples whose local distortion lies in the top quartile across all available points; the AUC reported for this group is computed using only the members and held-out samples that fall into this distortion range. As a baseline, we construct a "random" group by repeatedly (ten times) sampling equal numbers of members and non-members and reporting the mean AUC with one standard deviation (shown as a shaded band).

Across all datasets, we observe a consistent and pronounced ordering: higher-distortion quartiles yield substantially higher attack AUCs, indicating that regions of large decoder distortion exhibit stronger memorization. The magnitude of this effect varies by dataset. For CIFAR-10, the distortion gap between quartiles is large (Fig. 2), producing correspondingly large separations in attack performance; in contrast, CelebA exhibits nearly uniform distortion throughout its latent space, resulting in only minor differences in AUC across quartiles. We leave MNIST high-/low distortion samples in the appendix. Interestingly, the lines on ImageNet-1K experiments cross at one point. This weird crossing is dataset-dependent. Given that the earlier

time MIA is not trustworthy [21], it still shows a consistent phenomenon after passing the early time. This strong correlation between decoder's local distortion and memorization indicates that the choice of encoder–decoder architecture plays a central role in shaping the learned score field and, consequently, the memorization behavior of latent diffusion models.

4.3. Experiments on per-dimension memorization

Benchmark methods: To justify our work, we selectively choose four state-of-the-art threat models: SimA [21], SecMI [8], PIA [13], and Loss (also refered to "Naive" method) [19] as our benchmark methods acknowledging that list is not exhaustive. For each method, we followed the hyperparameter suggestion in their original paper. Notably, l_4-norm, l_2-norm, l_4-norm and l_2-norm were used for SimA, SecMI, PIA and Loss as suggested.

The complete results are reported in Tables 1 and 2. We report the best performance over probe time $t = [0 : 300]$. Throughout all experiments, we apply a dilution procedure, which removes the top-$k\%$ least influential latent dimensions from the attack vector prior to computing the norm. Although we do not conduct a search for the optimal value of k, we expect that such an optimal choice may vary across datasets and individual samples; exploring this dependency is an interesting direction for future work. For simplicity and consistency, we thus adopt $k = 40$ as a global setting. As a control, we also include a baseline that randomly discards 40% of dimensions when applying SimA (first row in each table), which typically reduces attack performance. In contrast, removing the least-memorizing dimensions based on our proposed per-dimension metric consistently improves all three evaluation measures across all methods and datasets. And from the "min δ", we found that Loss method usually achives the least improvement compared to others. These results indicate that latent-space diffusion models exhibit dimension-wise non-uniform memorization, with certain latent coordinates contributing disproportionately to overfitting and membership leakage.

4.4. Connection to high-frequency components

Lian et al. [15] reported that reconstructing images from denoised noise vectors and suppressing high-frequency components made members and non-members more separable. This approach performed well for diffusion models trained directly in the data domain, but in our experiments it reduced performance for more complex latent-space diffusion models (e.g., Stable Diffusion); see Section E in the appendix. We hypothesize that if the latent encoder were perfectly linear, then manipulating Fourier components would have behaved as expected in latent space. However, the nonlinear VAE encoder used in latent diffusion models does not simply map images to Fourier modes; instead, it performs

Table 3. Pearson correlation coefficients (r) between the decoder distortion and high/low-frequency amplitudes across datasets. The chosen threshold radius between low and high frequency for CIFAR-10 (32^2), CelebA (64^2) and ImageNet-1K (256^2) are 5, 10, 40, which are approximately 2.5% of the spectra for each.

Dataset	r(Distortion, LF)	r(Distortion, HF)
CIFAR-10	0.0814	0.7156
CelebA	-0.8713	-0.2865
ImageNet	0.1501	0.6440

denoising on a sub-manifold of the data domain for which the spectral analog is generally inaccessible (the spectrum of the Laplace-Beltrami operator).

To test this hypothesis, we computed the correlation between the local distortion of each image and the summed energy of its low-frequency and high-frequency spectra, for each of the three datasets CIFAR-10, CelebA, and ImageNet-1K. To select low frequencies, we summed the center elements of the Fourier transformed data; for CIFAR-10 (32^2), CelebA (64^2) and ImageNet-1K (256^2) the central radii summed were 5, 10, 40, which are approximately 2.5% of the spectra for each. The results, summarized in Table 3, show that at a high level the distortion reflects the intuition of Lian et al. [15], but that this analog is imperfect, especially for more structured datasets such as CelebA, where all data have aligned structure (spatially registered faces, in the case of CelebA).

5. Discussion

Our results suggest that membership inference and thereby model inversion for latent diffusion models may directly connected to the construction of the latent space itself. This runs counter to current thinking in attack methods, which apply the same methods to data-domain diffusion (e.g., the original DDPM model) and latent diffusion models. Instead, these results suggest that an optimal attack method would account for decoder geometry in its approach.

Distortion filtering uniformly improved performance across methods. This is surprising, not only because disparate approaches can be improved by the same influence measure filter, but also because the decoder from which the pullback metric distortion is computed is *not directly involved in the diffusion model fitting* itself. Common practice prescribes fitting a VAE to the dataset and then freezing the encoder/decoder pair, meaning the diffusion model fitting and any associated overfitting occurs on the latent space as generic data points; the diffusion model is not aware that it is operating on a latent code instead of data-domain points.

The necessity of accounting for decoder geometry for LDM inversion is therefore due to changes in distributional structures $p(z)$ versus $p(x)$, not diffusion dynamics. In the unlikely cases where a dataset had the same distribution as these VAEs, or an ill-fit VAE had the same distributional structure as the data domain, their diffusion models would have correspondingly matching dynamics. Because these cases are unlikely, in practice we conclude that inversion should account for decoder geometry, and more broadly that the overfitting phenomena in latent diffusion models is connected to the latent space.

Limitations and Caveats: We have tested only a subset of attack methods and a subset of possible datasets. Conditionalization, dataset size, and dataset structure clearly influence diffusion model behavior and thereby inversion dynamics. We can clearly design datasets where the data domain is similar to these latent spaces, and we can design trivial "VAEs" with no curvature. Our results are likely relevant for similar image data, which represent the mainstay of diffusion model applications, but they may not generalize to disparate domains (e.g., discrete diffusion on text or tabular data).

Crossover strata: The ImageNet subfigure of Figure 3 (at far right) show that the distortion strata lines cross over each other around timestep 70. This seems to inform upon interactions between the noise schedule and the length-scale of the latent space; clearly for faster noise schedules this crossover point would occur earlier. Interactions between noise schedules and decoder geometry may be important for attack parameter tuning and overall understanding of overfit phenomena.

6. Conclusion

We show that latent diffusion models exhibit spatially non-uniform memorization in latent space and demonstrate that this behavior is tightly correlated with the local distortion measured by the decoder pullback metric. We propose a principled, geometry-driven measure of per-dimensional memorization strength based on each latent dimension's contribution to decoder distortion. We provide extensive empirical evidence showing that removing less-memorizing latent dimensions of attack vector before computing the norm statistics consistently improves all metrics over all MIA methods.

These findings directly inform upon membership inference and model inversion in latent diffusion models, demonstrating the importance of the latent space and the decoder geometry to these problems.

References

[1] Stability AI. sd-vae-ft-mse. https://huggingface.co/stabilityai/sd-vae-ft-mse, 2023. Hugging Face model card; autoencoder (VAE) decoder finetuned on LM-Humans / LAION training set. 6

[2] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017. 1, 2, 4

[3] Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, 2020. 1, 2, 4

[4] 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)*, pages 5253–5270, 2023. 1, 3

[5] Clément Chadebec and Stéphanie Allassonnière. A geometric perspective on variational autoencoders. *Advances in neural information processing systems*, 35:19618–19630, 2022. 1, 2

[6] Hugging Face Community. Pokémon dreambooth dataset. https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions, 2022. Used for fine-tuning latent diffusion models. 6

[7] Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 30(7):1967–1974, 2018. 1

[8] 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*, pages 8717–8730. PMLR, 2023. 3, 4, 7

[9] Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. A probabilistic fluctuation based membership inference attack for diffusion models. *arXiv preprint arXiv:2308.12143*, 2023. 1, 3

[10] Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023. 1

[11] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017. 6

[12] Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. *arXiv preprint arXiv:2310.02557*, 2023. 2, 3

[13] Fei Kong, Jinhao Duan, RuiPeng Ma, Hengtao Shen, Xiaofeng Zhu, Xiaoshuang Shi, and Kaidi Xu. An efficient membership inference attack for the diffusion model by proximal initialization. *arXiv preprint arXiv:2305.18355*, 2023. 1, 3, 4, 7

[14] Qiao Li, Xiaomeng Fu, Xi Wang, Jin Liu, Xingyu Gao, Jiao Dai, and Jizhong Han. Unveiling structural memorization: Structural membership inference attack for text-to-image diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 10554–10562, 2024. 3, 2

[15] Puwei Lian, Yujun Cai, Songze Li, and Bingkun Bao. Unveiling impact of frequency components on membership inference attacks for diffusion models. *arXiv preprint arXiv:2505.20955*, 2025. 3, 7, 8, 1, 2

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015. 6

[18] Artem Lukoianov, Chenyang Yuan, Justin Solomon, and Vincent Sitzmann. Locality in image diffusion models emerges from data statistics. *arXiv preprint arXiv:2509.09672*, 2025. 2, 3

[19] Tomoya Matsumoto, Takayuki Miura, and Naoto Yanai. Membership inference attacks against diffusion models. In *2023 IEEE Security and Privacy Workshops (SPW)*, pages 77–83. IEEE, 2023. 1, 3, 4, 7

[20] Yan Pang, Tianhao Wang, Xuhui Kang, Mengdi Huai, and Yang Zhang. White-box membership inference attacks against diffusion models. *arXiv preprint arXiv:2308.06405*, 2023. 3

[21] Mingxing Rao, Bowen Qu, and Daniel Moyer. Score-based membership inference on diffusion models. *arXiv preprint arXiv:2509.25003*, 2025. 1, 4, 7

[22] 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*, pages 10684–10695, 2022. 1, 2, 6

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 6

[24] Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 315–323, 2018. 2, 4

[25] Minglei Shi, Haolin Wang, Wenzhao Zheng, Ziyang Yuan, Xiaoshi Wu, Xintao Wang, Pengfei Wan, Jie Zhou, and Jiwen Lu. Latent diffusion model without variational autoencoder. *arXiv preprint arXiv:2510.15301*, 2025. 3

[26] 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*, pages 6048–6058, 2023. 1

[27] 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, 2023. 1

[28] Binxu Wang and John J Vastola. The hidden linear structure in score-based models and its application. *arXiv preprint arXiv:2311.10892*, 2023. 2, 3

[29] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descrip-

tions. *Transactions of the association for computational linguistics*, 2:67–78, 2014. 6

[30] 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. 1, 3

[31] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025. 3

[32] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016. 1

Latent Diffusion Inversion Requires Understanding the Latent Space

Supplementary Material

A. Approximation of Local Distortion

An pseudo-code to efficiently compute local distortion using randomized singular value decomposition is provided in Algo. 1. Noticeably, this code computing Jacobian–vector products using jvp/vjp packages in pytoch without explicitly computing the Jacobian matrix. In our experiments, finite-difference is also a good approximation of jvp when automatic differentiation is unavailable

---

Algorithm 1 Matrix–Free Randomized SVD for Top-k Singular Values of $J_D(z)$

---
*[t]

Require: Decoder mean map $D : \mathcal{Z} \to \mathcal{X}$; latent point $z \in \mathcal{Z}$; target rank k; oversampling p; power passes q; flag USE_JVP∈ {TRUE, FALSE}; finite-difference step h

Ensure: Approx. top-k singular values $\{s_i\}_{i=1}^{k}$ of $J_D(z)$ and log–volume $\sum_{i=1}^{k} \log s_i$

1: $d \leftarrow \dim(\mathcal{Z}), \quad m \leftarrow \dim(\mathcal{X}), \quad \ell \leftarrow \min(k+p, d)$
2: **define** decoder–Jacobian oracles at z:
   $\mathcal{J}(V) = J_D(z)V \in \mathbb{R}^{m \times r}$ **via** JVP if USE_JVP=TRUE,
   else central differences: $[(D(z+hv_j) - D(z-hv_j))/(2h)]_j$
   $\mathcal{J}^\top(Y) = J_D(z)^\top Y \in \mathbb{R}^{d \times r}$ **via** VJP (reverse-mode autodiff)
3: Draw $V_0 \sim \mathcal{N}(0, I_{d \times \ell}); \quad V \leftarrow \text{qr}(V_0)$ ▷ $V \in \mathbb{R}^{d \times \ell}$
4: **for** $t = 1$ to q **do** ▷ optional power iteration on $G = J_D^\top J_D$
5: $\quad Y \leftarrow \mathcal{J}(V) \in \mathbb{R}^{m \times \ell}; \quad G \leftarrow \mathcal{J}^\top(Y) \in \mathbb{R}^{d \times \ell}$
6: $\quad V \leftarrow \text{qr}(G)$
7: **end for**
8: $Y \leftarrow \mathcal{J}(V) \in \mathbb{R}^{m \times \ell}; \quad Q \leftarrow \text{qr}(Y) \in \mathbb{R}^{m \times \ell}$
9: $T \leftarrow \mathcal{J}^\top(Q) \in \mathbb{R}^{d \times \ell}$
10: $\{\hat{s}_i\}_{i=1}^{\ell} \leftarrow \text{svdvals}(T)$; **sort** $\hat{s}_i$ in descending order
11: $s_i \leftarrow \hat{s}_i$ for $i = 1, \ldots, k$ ▷ top-k singular values of $J_D(z)$
12: **return** $\{s_i\}_{i=1}^{k}, \quad \text{log–volume} = \sum_{i=1}^{k} \log s_i$

B. Plots of attack score versus local distortion

This is repetitive content. We keep it here to maintain the indexing consistency between the main text and the appendix. It will be deleted in future versions.

C. Datasests and splits

The dataset and member/non-member splits information are provided in Table 4.

D. Per–Dimension Influence via Hutchinson Estimator

A pseudocode of our method to compute the per-dimension influence on memorization via the Hutchinson Estimator is

---

Algorithm 2 Per–Dimension Influence via Hutchinson Estimator

Require: Decoder D, latent samples $\{z^{(n)}\}_{n=1}^{N}$ with $z^{(n)} \in \mathbb{R}^d$, number of Monte Carlo probes n_{mc}, stability constant $\varepsilon > 0$

Ensure: Matrix PerDim $\in \mathbb{R}^{N \times d}$ containing per–dimension log–influence for each $z^{(n)}$

1: **for** $n = 1$ to N **do** ▷ Process each latent code independently
2: $\quad z \leftarrow z^{(n)}$; enable gradient on z
3: $\quad x \leftarrow D(z); \quad x \in \mathbb{R}^m$ ▷ Decoder output (flattened)
4: $\quad$ diag_est $\leftarrow \mathbf{0} \in \mathbb{R}^d$ ▷ Accumulator for $\text{diag}(J_D^\top J_D)$
5: $\quad$ **for** $j = 1$ to n_{mc} **do** ▷ Hutchinson probes
6: $\quad\quad v \sim \mathcal{N}(\mathbf{0}, I_m)$ ▷ Random probe in output space
7: $\quad\quad g \leftarrow J_D(z)^\top v$ ▷ Compute VJP via reverse–mode autodiff
8: $\quad\quad$ diag_est $\leftarrow$ diag_est $+ g \odot g$ ▷ Elementwise square and accumulate
9: $\quad$ **end for**
10: $\quad$ diag_est $\leftarrow$ diag_est$/n_{\text{mc}}$ ▷ Monte Carlo average
11: $\quad$ per_dim_log $\leftarrow \frac{1}{2} \log(\text{diag\_est} + \varepsilon)$ ▷ Log–compressed influence
12: $\quad$ PerDim$[n, :] \leftarrow$ per_dim_log
13: **end for**
14: **return** PerDim

---

provided in Algo. 2.

E. Latent diffusion attack by smoothing high-frequency components

In this section, we apply the method proposed by Lian et al. [15] to latent diffusion models (LDMs), specifically Stable Diffusion. Although the method was originally developed for data-domain diffusion models, we extend it to the latent domain using the following formulation.

Loss: The attack statistics can be expressed as, for $\varepsilon \sim \mathcal{N}(0, I)$:

$$\text{Loss} = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \, t)\|, \quad (10)$$

where $\epsilon_\theta(\cdot)$ indicates the predicted noise. According to Appendix B of Lian et al. [15]. We derive

$$z_0^{\text{target}} = \frac{z_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon}{\sqrt{\bar{\alpha}_t}}, \quad (11)$$

$$z_0 = \frac{z_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(\sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \, t)}{\sqrt{\bar{\alpha}_t}}. \quad (12)$$

To smooth the high-frequency components, we should map latent codes z to the image x. Then, $x_0 = D(z_0)$ and $x_0^{\text{target}} = D(z_0^{\text{target}})$.

Table 4. Datasets and splits used for our experiments. **VAE** denotes the **VAE of LDMs**. **LM** denotes **Diffusion Part of LDMs**

Model	Member	Held-out	Pre-trained (VAE)	Pre-trained(LM)	Splits	Resolution	Cond.
Latent Diffusion Models	MNIST	MNIST	No	No	30k/30k	28	–
	CIFAR-10	CIFAR-10	No	No	25k/25k	32	–
	CelebA	CelebA	No	No	30k/30k	64	–
	ImageNet-1k(train)	ImageNet-1K(Val)	Yes	No	100k/100k	256	class
Stable Diffusion V1-4	Pokémon	Pokémon	Yes	Yes	416/417	512	text
	COCO2017-Val	COCO2017-Val	Yes	Yes	2.5k/2.5k	512	text
	Flickr30k	Flickr30k	Yes	Yes	10k/10k	512	text

PIA: Similarly, it can be expressed as:

$$\text{PIA} = \left\| \epsilon_\theta(z_0, 0) - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(z_0, 0),\, t\big) \right\|, \tag{13}$$

where $\epsilon_\theta(z_0, 0)$ represents the noise prediction for z_0. According to Appendix B of Lian et al. [15], it can be converted to

$$z_0^{\text{target}} = \frac{z_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(z_0, 0)}{\sqrt{\bar{\alpha}_t}}, \tag{14}$$

$$z_0 = \frac{z_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(\sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(z_0, 0),\, t)}{\sqrt{\bar{\alpha}_t}}. \tag{15}$$

Likewise, $x_0 = D(z_0)$ and $x_0^{\text{target}} = D(z_0^{\text{target}})$.

A high-frequncy filter is defined the same as Li et al. [14],

$$\mathcal{F}(x) = \text{IFFT}(\text{FFT}(x) \odot \beta(r)), \tag{16}$$

where FFT and IFFT denotes the *Fast Fourier Transform* and *Inverse Fast Fourier Transform*. $\odot$ denotes element-wise multiplication, and $\beta_{i,t}(r)$ is a mask that keeps the low-frequency components and smooths the high-frequency components. Specifically,

$$\beta(r) = \begin{cases} s, & \text{if } r > r_t, \\ 1, & \text{otherwise.} \end{cases} \tag{17}$$

where s is frequency-dependent scaling factor, r denotes the radius of frequency-domain, and r_t is the high-frequency threshold radius. Then the attack statistics for both Loss and PIA methods become:

$$\|\mathcal{F}(x_0) - \mathcal{F}(x_0^{\text{target}})\|. \tag{18}$$

Under the assumption that latent-space diffusion models inherit the same learning dynamics as data-domain diffusion models, one would expect the proposed high-frequency filtering technique to be equally effective in both settings. Contrary to this expectation, our experiments show that the method decreases the attack performance for Stable Diffusion (Table 5). This divergence suggests that the mapping from the data domain to the latent domain—typically implemented through a nonlinear VAE encoder—introduces non-trivial distortions that alter the frequency structure relevant for membership inference. Consequently, careful consideration of this domain transformation is essential when adapting techniques developed for data-domain models.

F. More image examples on different local distortion

More examples of images in the high (low) distortion region are provided in Fig. 4.

G. Architecture details of all LDM instances

CIFAR-10. We train a convolutional VAE at 32×32 RGB resolution with 3 input channels, base width 128, and a 4-channel latent space. The encoder consists of strided convolutions and residual blocks with GroupNorm and SiLU activations, downsampling $32 \to 16 \to 8$; the decoder mirrors this structure. The VAE is optimized for 120 epochs on the member subset using AdamW (learning rate 2×10^{-4}, $\beta = (0.9, 0.999)$, weight decay 10^{-4}), with an ℓ_1 reconstruction term and a KL divergence term weighted by $\beta_{\text{KL}} = 10^{-2}$. On the frozen latent space, we train a DDPM-style UNet with 128 base channels, channel multipliers $(1, 2, 2, 2)$, two residual blocks per resolution level, dropout 0.1, and GroupNorm. The diffusion model uses a linear noise schedule with $T = 1000$ steps and $\beta \in [10^{-4}, 2 \times 10^{-2}]$, and is trained with AdamW (same hyperparameters) for 2048 epochs with batch size 128.

CelebA. For CelebA, we first center-crop faces to 140×140, resize to 64×64, and apply random horizontal flips and per-channel normalization to $[-1, 1]$. We train a VAE with the same architectural configuration (3 input channels, base width 128, 4-channel latent space) for 120 epochs using AdamW (learning rate 2×10^{-4}, $\beta = (0.9, 0.999)$, weight decay 10^{-4}), but with a smaller KL weight $\beta_{\text{KL}} = 10^{-3}$. The latent diffusion model is a UNet with 224 base channels and channel multipliers $(1, 2, 3, 4)$, two residual blocks per level, dropout 0.1, and a linear noise schedule with $T = 1000$ diffusion steps and $\beta \in [10^{-4}, 2 \times 10^{-2}]$. It is trained with AdamW (same optimizer settings) for 512 epochs with batch size 256.

Table 5. Attack performance of four baseline methods used to evaluate memorization of **Stable Diffusion**. Best results in **bold**.

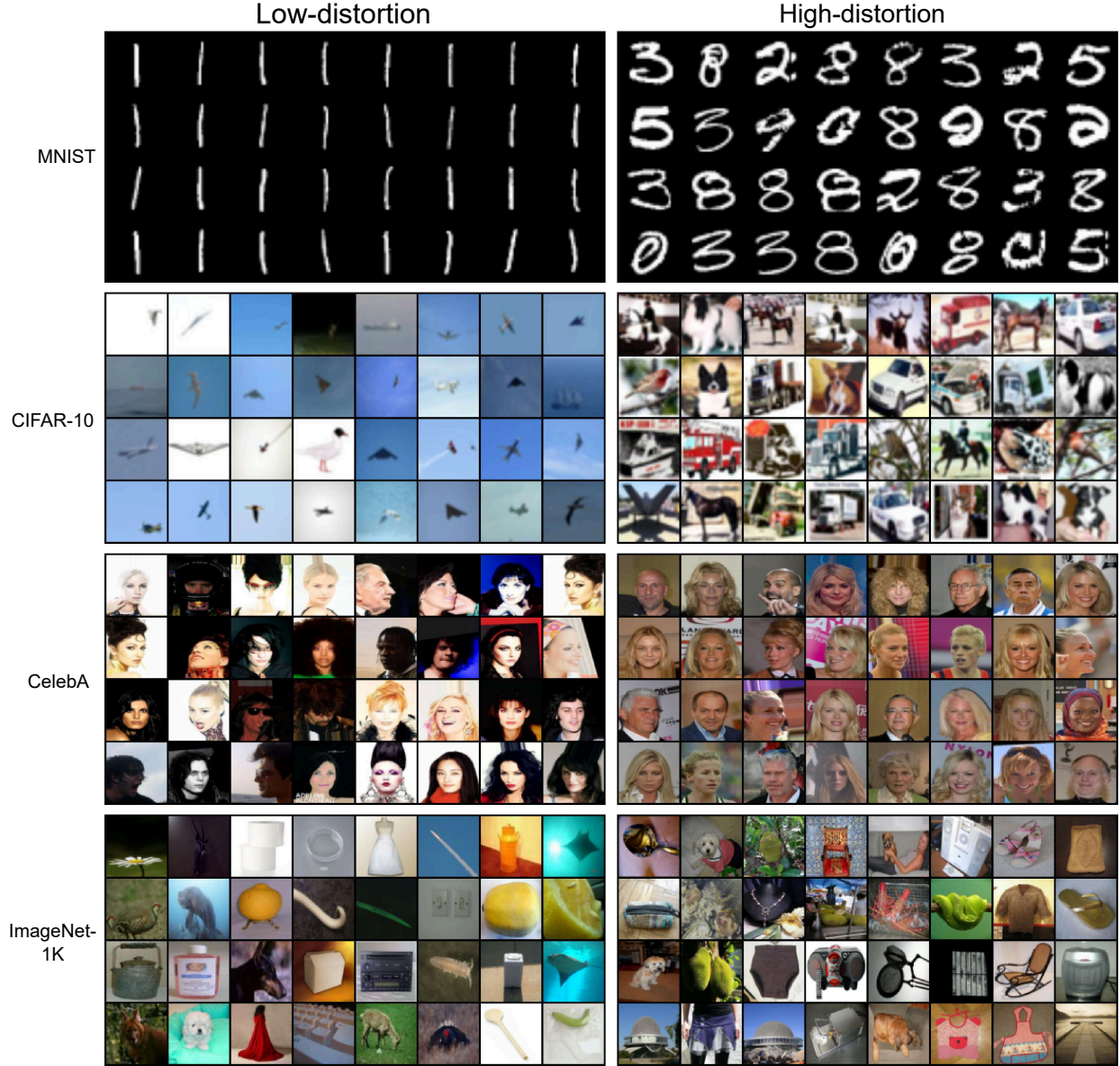| Method	Pokémon (%)			MS-COCO (%)			Flickr (%)		
	AUC↑	ASR↑	TPR@1%FPR↑	AUC↑	ASR↑	TPR@1%FPR↑	AUC↑	ASR↑	TPR@1%FPR↑
Loss	92.03	85.47	40.77	83.26	75.80	13.48	63.13	59.74	2.35
Loss (+high freq. filter)	84.39	78.40	19.90	73.46	68.08	4.32	57.52	55.81	1.61
PIA	94.93	90.52	32.85	92.51	85.72	24.76	68.88	64.61	2.67
PIA (+high freq. filter)	89.81	84.27	19.90	73.32	67.72	4.96	57.37	55.61	1.43

Figure 4. More examples of images in high (low) distortion region

ImageNet-1K. We train a class-conditional latent diffusion model on ImageNet-1K using the pretrained Stable Diffusion VAE (`sd-vae-ft-mse`) as a frozen encoder–decoder (with default $\beta_{KL} = 1e - 6$). Input images are

resized and center-cropped to 256×256 RGB, augmented with random horizontal flips, and normalized to $[-1, 1]$. The VAE deterministically encodes images into $4 \times 32 \times 32$ latents ($8\times$ spatial downsampling), which are scaled by 0.18215 before diffusion and rescaled back when decoding.

On this latent space, we train a UNet with 4 input/output channels, 2 residual blocks per resolution, and block widths $(192, 384, 576, 960)$, using cross-attention layers with a learned class embedding of dimension 512. Class conditioning is implemented via an embedding table over the 1,000 ImageNet labels, passed as the UNet cross-attention context. The diffusion process uses $T = 1000$ steps with a linear noise schedule $\beta \in [1.5 \times 10^{-3}, 1.95 \times 10^{-2}]$.

We construct a balanced training subset with 50 images per class (50k images total) from the ImageNet-1K training split and use the standard validation split for evaluation and qualitative sampling. The UNet is optimized for 600 epochs with AdamW (learning rate 10^{-4}, $\beta = (0.9, 0.999)$, zero weight decay) and a cosine learning-rate schedule with 1,000 warm-up steps. Training is performed with mixed-precision (FP16) and distributed data parallelism with an effective per-device batch size of 21.

Fine-tuning on Stable diffusion. We reuse the Stable Diffusion V1-4 architecture for all stable diffusion experiments.