LOSS CURVATURE AS A SAMPLING-FREE PROXY FOR MEMORIZATION IN DIFFUSION MODELS

Shashank Sule[*,1] **Xinyue Yu**[*,1] **Boonam Shin**[2]

Yuxuan Liu[1] **Abolfazl Hashemi**[2] **Hayden Schaeffer**[1]

ABSTRACT

Understanding the mechanism behind why diffusion models memorize their training data remains an open question in generative modeling. In this paper, we examine this question from two perspectives: (a) the model-level perspective concerning the onset of memorization in a model during training, and (b) the sample-level perspective concerning the proclivity of an individual sample to be memorized by a given model. We show that the so-called input loss curvature, or the Frobenius norm of the Hessian of the diffusion model's training loss, is an effective metric for quantifying both perspectives on memorization. In particular, we empirically demonstrate that input curvatures averaged across training and validation samples can identify when a model is likely to start regurgitating its training data when tracked over training time. However, this metric is not sufficient for measuring an individual sample's likelihood of being memorized. To address this perspective, we introduce a counterfactual definition of memorization in diffusion models, extending the analogous notion from supervised learning proposed by [Feldman, 2019]. We prove that this counterfactual memorization is controlled by the input curvature of the leave-one-out model, thus establishing a direct theoretical link between a sample's local loss landscape and its generation likelihood. We validate our derived bounds through several empirical experiments and show that input-dependent curvature provides a proxy for memorization.

1 Introduction

Diffusion and score-based models are state-of-the-art generative models across domains from vision to drug discovery [SDWMG15, HJA20, SE19, SSDK+21, RBL+22, ND21, AAD+24, RPG+21]. Their appeal lies in generating novel yet realistic data, suggesting they estimate the data distribution from finitely many samples. Yet diffusion models sometimes regurgitate training samples, a failure mode called *memorization*, which poses privacy and copyright risks [CHN+23, SSG+23a, SSG+23b, GDP+23]. Existing work studies memorization at two levels:

1. *Model-level memorization* asks how closely the trained score matches the minimizer of the empirical denoising score matching loss [Vin11, BDK+25], namely the score of the Gaussian KDE of the training set, whose sampling reproduces the training data exactly. Tracking training time τ, models first generalize (at τ_{gen}) and later memorize (at τ_{mem}) [BM23, BBdBM24, YCKR23, AVS+24], but these time-scales are known only asymptotically, and measuring the memorized fraction $f_{mem}(\tau)$ requires costly sampling.
2. *Sample-level memorization* asks which training points are copied. Previous work attributes sample-level memorization to the presence of duplicates [CHN+23, SSG+23b, ZIL+23, CTW+21]. More generally, sample-level memorization refers to the influence of an individual sample on a model. This connects to to the long-tail theory of [Fel20], which motivates the counterfactual memorization measure of [FZ20] (see also [ZBH+17]). Extending this measure to diffusion models is nontrivial because the training objective (denoising loss) differs from the test objective (likelihood), and generalization in diffusion models has its own geometric structure [KGSM24].

[*]Equal contribution.

[1]Department of Mathematics, University of California, Los Angeles, CA 90095, USA.

[2]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA.

These perspectives motivate *sampling-free* proxies of memorization which can ideally be measured during training. We propose using *input loss curvature* from [RSR24] given by

$$\mathrm{Curv}_\phi(x) = \mathbb{E}_{\phi\sim\mathcal{A}(S)}\left[\|\nabla_x^2 \ell(\phi, x)\|_F^2\right], \tag{1}$$

as such a proxy. We use it for two reasons: (i) it is the leading order term that bounds Feldman's memorization score, is among the strongest membership-inference signals, and estimates differential privacy [RSHR24, RSR24, GRR24, SSSS17, CCN+22, DMNS06]; (ii) the Hessian of the potential $-\log p(x)$ governs the local geometry of sampling, with higher curvature confining Langevin trajectories near a minimizer for longer [BdH15, Ber13]. However, during training it is prohibitively expensive to compute these negative log likelihoods–instead, we investigate whether the second order information on the loss can help predict the properties implied by the second order information of the negative log likelihood.

Contributions.

(a) *Curvature over training image tracks memorization.* We demonstrate that on ImageNet 64×64 and CelebA 32×32, $\mathrm{Curv}_\phi(x)$ averaged over the training set corresponds to the f_{mem} score and rises as the model overfits. Moreover, at the onset of memorization, samples that are copied (i.e appear to be in the training set) have higher curvature than samples that are not copied.

(b) *Held-out curvature separates from training curvature during memorization.* While curvature over training images rises, the curvature over the validation set starts separating from the training curvature at the onset of memorization. Surprisingly, validation curvature hits a minimum during training and starts rising as memorization proceeds, while validation loss continues to decrease. The local minimum of validation curvature provides an early-stopping measure for diffusion models that indicates the prevalence of model-level memorization.

(c) *A counterfactual notion of memorization.* We illustrate that while curvature indicates model-level memorization, it cannot signal the copy frequency of individual samples. To quantify such individual sample influence, we introduce a new notion of memorization, termed counterfactual memorization given by the log-odds of sampling x_i with and without its inclusion in the training data.

(d) *Counterfactual memorization is bounded by curvature.* By lifting the differences in log-likelihoods to model losses via a decomposition given in [HLC21], we quantify the influence of training dynamics on counterfactual memorization. In particular, we demonstrate that when the model starts overfitting, the log-likelihood difference is captured at the leading order by the curvature of the leave-one-out model and the nearest neighbour distance.

2 Background

2.1 Diffusion Models

A diffusion model is a generative algorithm for sampling from an unknown density p by transforming noise into clean data. To learn such a model we first establish the following forward process:

$$dY_t = f(t, Y_t), dt + g(t), dW_t, \qquad Y_0 = x \sim p, \tag{2}$$

When $f(t, y) = f(t)y$, the transition kernel is Gaussian: $Y_t \mid Y_0 = x \sim \mathcal{N}(m(t)x, \sigma(t)^2 I)$. The schedules (m, σ) and the coefficients (f, g) thus determine each other:

$$f(t) = \frac{m'(t)}{m(t)}, \qquad g(t)^2 = \frac{d}{dt}\sigma(t)^2 - 2\frac{m'(t)}{m(t)}, \sigma(t)^2. \tag{3}$$

Anderson's theorem [And82] gives the backward SDE, whose law at time t is that of Y_{T-t}:

$$d\overline{X}_t = \left(\bar{g}(t)^2 \nabla \log p_{T-t}(\overline{X}_t) - \bar{f}(t, \overline{X}_t)\right), dt \tag{4}$$

$$+ \bar{g}(t), dB_t \tag{5}$$

where $\bar{f}(t, x) = f(T - t, x)$ and $\bar{g}(t) = g(T - t)$. Thus, given f and g, sampling from p reduces to knowing the score $\nabla \log p_t$ and running the backward SDE from $t = 0$ to $t = T$. Via the Tweedie-Miyasawa formula

$$\mathbb{E}[X \mid Y_t = y] = \frac{y + \sigma^2(t)\nabla \log p(t, y)}{m(t)}, \tag{6}$$

we can instead parameterize the denoiser $\mathbb{E}[X \mid Y_t = y]$ and recover the score by rearranging (6). We therefore train a model D_ϕ by MMSE denoising:

$$\ell(\phi, x) := \mathbb{E}_{t\in[0,T], y_t\sim Y_t}\left[\lambda(t)\|x - D_\phi(y_t, t)]\|_2^2 \mid Y_0 = x\right]. \tag{7}$$

Here $\lambda(t)$ is a choice of weighting function. Substituting the learned score $s_\phi = \sigma(t)^{-2}(m(t)D_\phi - y)$ gives the *plug-in SDE*:

$$dX_t = (\overline{g}(t)^2 s(t, X_t) - \overline{f(t, X_t)})\, dt + \overline{g}(t)\, dB_t. \tag{8}$$

The plug-in SDE has density $p_\phi(y, t)$. At generation time we integrate (8) forward, obtaining the terminal distribution $p_\phi(\cdot, T)$ as an approximation to the true distribution p.

2.2 DDPMs

In this paper, we use Denoising Diffusion Probabilistic Models (DDPMs) [HJA20] corresponding to taking $f(t) = -\frac{1}{2}\beta(t)$ and $g(t) = \sqrt{\beta(t)}$ for some non-negative rate function β. Writing $\bar{\alpha}(t) := \exp\left(-\int_0^t \beta(u)\, du\right)$, one finds $\Phi(t) = \sqrt{\bar{\alpha}(t)}$ and, using $\frac{d}{ds}\bar{\alpha}(s)^{-1} = \beta(s)\bar{\alpha}(s)^{-1}$,

$$m(t) = \sqrt{\bar{\alpha}(t)}, \qquad \sigma(t)^2 = 1 - \bar{\alpha}(t). \tag{9}$$

To sample from DDPMs, we use an accelerated non-Markovian sampler given in [SME21]. More generally, note that the sampling process via (8) is implemented by discretizing the SDE via Euler-Maruyama, leading to a stochastic sampling trajectory. A deterministic sampling, using the probability flow ODE or flow-matching may also be used [SSDK+21, LCBH+23]. In Appendix B, we provide experimental details and additional details on flow matching.

2.3 Log-Likelihoods via the Denoising Loss

Diffusion models are trained by using the loss (7). However, at test time, their performance is measured by the fidelity of $p(\cdot, T)$ to the true distribution p. Therefore, their training and test tasks are rather different. However, an evidence lower bound (ELBO) shows that the denoising loss bounds the negative log-likelihood of the distribution $p_\phi(\cdot, T)$. In fact, [HLC21] described the gap between the ELBO and $-\log p_\phi(\cdot, T)$ exactly under the likelihood weighting $\lambda(t) = g^2(t)$:

Proposition 2.1 (Theorem 4 in [HLC21])**.** *Let $\ell(\phi, x)$ be the denoising loss (7), $p_\phi(x, T)$ the distribution of the plug-in SDE (8) at time T and s_ϕ the learned score function in (8). Then*

$$\begin{aligned} &-\log p_\phi(x, T) - \ell(\phi, x) = C \\ &+ \underbrace{\int_0^T \mathbb{E}\left[g^2(t)\|s_\phi(Y_t, t) - \nabla \log_\phi(Y_t, T-t)\|_2^2 \mid Y_0 = x\right]\, dt}_{Gap_\phi(x)}, \end{aligned} \tag{10}$$

where C is a constant independent of x or ϕ but dependent on the data dimension d, and the noise schedules f, g.

The RHS in (10) is purely a *self-consistency* gap, i.e it measures how close the score of the reversed plug-in SDE is to the original forward SDE.

2.4 Memorization

Here we summarize the framework of [Fel20, FZ20]. Let $\mathcal{A}$ be a training algorithm, possibly randomized, and write h_S^ϕ for the model with parameters $\phi \sim \mathcal{A}(S)$. Let $\ell(h_S^\phi, z)$ be its loss on an example z, and let $S^{\setminus i} = S \setminus z_i$. The memorization of z_i is the change in expected loss at z_i caused by its inclusion in the training set,

$$M(z_i, S) = \Big|, \mathbb{E}_{\phi \sim \mathcal{A}(S)}[\ell(h_S^\phi, z_i)] \; - \mathbb{E}_{\phi' \sim \mathcal{A}(S^{\setminus i})}[\ell(h_{S^{\setminus i}}^{\phi'}, z_i)]\Big|.$$

A large value means the model only fits z_i because it was trained on z_i.

3 Empirical Results

We begin with an empirical investigation of curvature as a proxy for memorization in diffusion models. To do so, we train DDPMs to sample two datasets: (1) 64×64 RGB ImageNet-1K [DDS+09, RDS+15], and (2) 32×32 greyscale CelebA images, also used in [BM23]). To further isolate the memorization phenomenon, we train six distinct DDPMs on datasets of size $n = 2^k$ for $7 \le k \le 12$ for ImageNet-1k and two distinct DDPMs on datasets of size $n = 128, 10240$ for CelebA, noting that for CelebA, $n = 128$ corresponds to a memorizing model while $n = 10240$ corresponds to a

generalizing model. Across the training time τ, we track the mean curvature over the training samples and validation samples:

$$\text{Avg. train curvature} = \mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2], \tag{11}$$

$$\text{Avg. val curvature} = \mathbb{E}_{x\sim\text{val.}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2]. \tag{12}$$

We also generate $10,000$ samples to compute the f_{mem} score. Following [GDP+23, YCKR23, BUBM25], a generated sample x is considered memorized if

$$\left[\frac{\|x - a^{\mu_1}\|_2}{\|x - a^{\mu_2}\|_2} < k\right], \tag{13}$$

where a^{μ_1} and a^{μ_2} are the nearest and second nearest neighbors in the training set to x. We say that a training sample x_i is *copied* if it appears in the generated set. Its *copy count* $F(i)$ refers to the frequency of occurrences of x_i in the generations. In Appendix C we visualize the samples generated from our various models. Here we assess the input loss curvature of our models during training.

Evolution of input curvature during training. In Figure 1, we report our experiments on CelebA. We find that:

1. Mean training curvature rises over training together with $f_{\text{mem}}(\tau)$ (panel (C)). The rise alone does not mark the onset: training curvature is already increasing while f_{mem} is still flat.
2. The onset shows up in the validation curvature. It stays with the training curvature and then diverges, in both panels (A) and (B), at the step where $f_{\text{mem}}(\tau)$ starts to accelerate. The same split appears for the generalizing model (panel (D)), where memorization is delayed until $\tau \approx 10^5$. This motivates defining the relative change in avg. train and validation curvatures as a stopping criterion:

$$\Delta\text{Curv}(\tau) := \frac{|\text{Avg. train curvature} - \text{Avg. val. curvature}|}{|\text{Avg. train curvature}|} \tag{14}$$

   we may consider stopping training when $\Delta\text{Curv}(\tau) \geq \varepsilon$, where ε modulates the threshold required to limit memorization. Notably this estimator *does not need* to generate any samples during training, saving significant computational cost.
3. Once memorization is heavy, every training image is reproduced but at different frequencies. However, the curvature of an individual image does not predict its copy count (Panel (F)).

In Figure 2, we repeat these experiments for ImageNet, observing patterns consistent with CelebA. In Panel (F), we confirm that $f_{mem}(\tau)$ rises and reaches 100% by $\tau \approx 10^6$ for every model except $n = 4096$. The validation loss, a zeroth-order quantity, falls throughout training for every n (Panel (E)), so the held-out loss alone does not detect memorization. However, in panel (A), we find that input curvature for training samples rises with τ, but the average validation curvature *reaches a local minimum* and then rises once the model begins to memorize heavily. In Panel (D), we show that $\Delta\text{Curv}(\tau)$ reaches a local maximum at around $\tau = 10^5$, where most models are memorizing.

Appendices D and E provide additional numerical experiments.

4 Sample-Level Memorization

Our results in Section 3 show that while curvature, aggregated over either training or held-out sets, is a good indicator of model-level memorization, it is not a good signal for whether a given sample will be memorized more than another one. To address this gap, we introduce a counterfactual memorization, analogous to what was defined in [ZIL+23]. Before we begin, we establish the relevant notation.

4.1 Notation

The notation will be borrowed from [RSHR24]. Let $\mathcal{X}$ be the space of inputs, $\mathcal{Y}$ the space of outputs, and h a model $h : \mathcal{X} \to \mathcal{Y}$. The joint distribution in the input-output pair $z = (x, y)$ will be denoted p_Z and its X and Y marginals will be the input and output distributions. The model will be drawn from some parametric function class $\mathcal{C}$ parameterized by $\Phi \subseteq \mathbb{R}^P$ where P represents the total number of parameters. To mark its dependence on parameters, the model will therefore be denoted as h^ϕ. In practice, we draw a finite number of samples $S = \{z_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{S}^n = (\mathcal{X} \times \mathcal{Y})^n$ and *train* to fit a loss $c : \mathcal{Y} \times \mathcal{Y}'$ on the data. We assume the n-data points are sampled i.i.d; therefore p_Z induces a distribution (via independence) on $\mathcal{S}^n$, termed p_Z^n. Given $S \sim p_Z^n$, the training process is quantified by

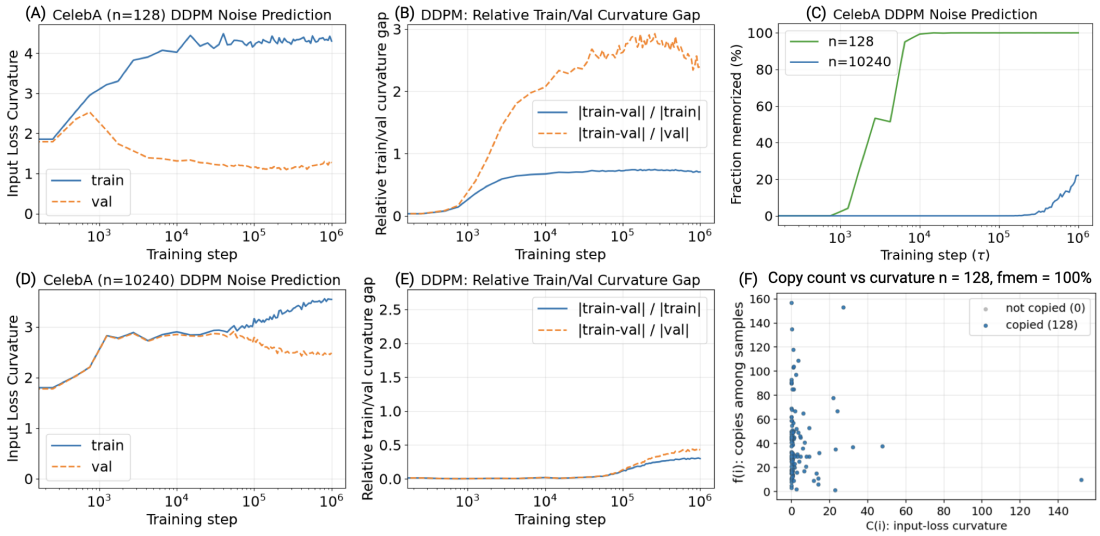


Figure 1: (A) Train vs. validation input curvature for CelebA, n=128, (B) Val. curvature starts to diverge from train curvature when $f_{mem}(\tau)$ starts accelerating, (C) $f_{mem}(\tau)$ values for $n = 128, 10240$ on CelebA, (D) and (E) Same experiment for CelebA, n=10240. Note that this model undergoes generalization, and memorization sets in much later at $\tau \approx 10^5$, (F) Curvature of an individual sample is not a good predictor of its copy count.

some algorithm $\mathcal{A} : \mathcal{S}^n \to \mathcal{M}$, where $\mathcal{M} \subseteq \mathcal{P}_2(\Phi)$ and $\mathcal{P}_2(\Phi)$ is the class of square integrable random m on Φ. A model trained with data S with parameters ϕ is thus denoted h_S^ϕ. We may define the loss over the model h_S^ϕ and z as

$$\ell(h_S^\phi, z) = c(h_S^\phi(x), y). \tag{15}$$

Wherever necessary we will abuse notation and identify the function $\phi, z \to \ell(h_S^\phi, z)$ with ℓ. Given a dataset S we will also define the losses on S as follows:

$$\ell(\phi, S) := \frac{1}{|S|} \sum_{i=1}^{|S|} \ell(h_S^\phi, z_i). \tag{16}$$

We also set $\ell^*(S) := \inf_{\phi \in \Phi} \ell(\phi, S)$.

4.2 The Counterfactual Log-Odds of Sampling

We may simplify the notation above in the specific case of diffusion models. In particular, the training objective (7) is *unsupervised*, and thus $z_i = x_i$. Moreover, we assume h^ϕ is produced via score matching/denoising and that it is then used to drive the SDE (8). We now define the counterfactual memorization:

Definition 4.1. *Let $z_i \in S$ be a training example and $\phi \sim \mathcal{A}(S), \phi' \sim \mathcal{A}(S^{\setminus i})$ be parameter distributions after optimization on the datasets $S, S^{\setminus i}$. Let $p_\phi(\cdot, T), p_{\phi'}(\cdot, T)$ be the time marginals of the distributions produced by driving the plug-in SDE (8). Then the* sampling time memorization *of z_i is given by:*

$$m(z_i, S) := \mathbb{E}_{\phi,\phi'}[\log p_\phi(z_i, T) - \log p_{\phi'}(z_i, T)]. \tag{17}$$

Note that $m(z_i, S)$ represents the log-odds of sampling z_i with and without its presence in the training data. Large m implies that the model would be much less likely to sample z_i if it wasn't included in the dataset. We also introduce a difference in the counterfactual *losses*, borrowed from [RSR24, RSHR24, Fel20, FZ20]:

$$M(z_i, S) = \mathbb{E}_{\phi \sim \mathcal{A}(S)}[\ell(h_S^\phi, z_i)] \tag{18}$$

$$- \mathbb{E}_{\phi' \sim A(S^{\setminus i})}[\ell(h_{S^{\setminus i}}^{\phi'}, z_i)]. \tag{19}$$

We now use the decomposition (10) to write m in terms of M:

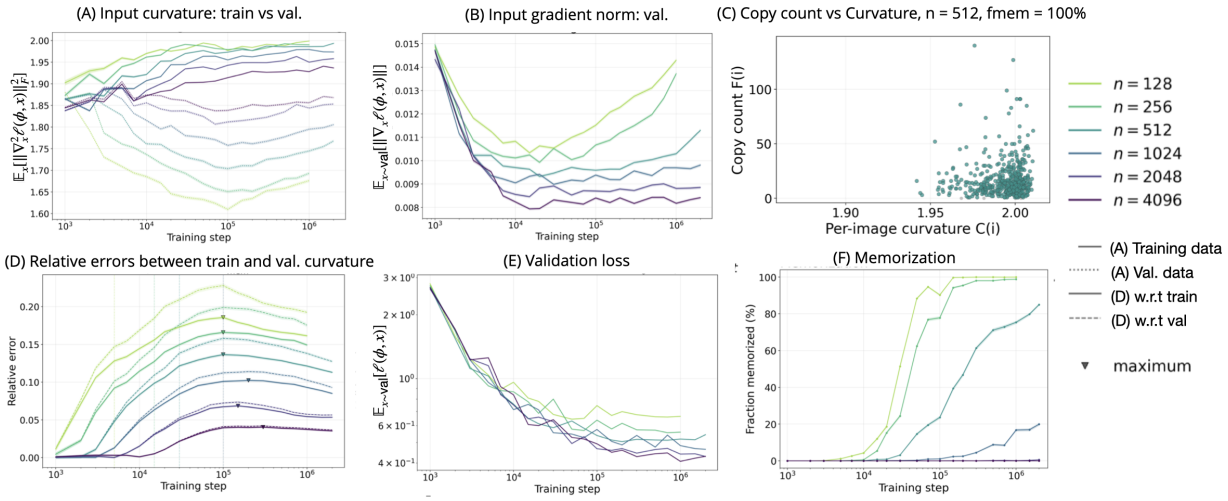


Figure 2: (A) Input (11) curvatures consistently rise but validation (12) curvatures reach a local minimum at approximately peak memorization for DDPMs on different data sizes, (B) Sizes of gradients of validation loss also display local minima, (C) Input curvature on training data tracks the largest frequency of memorized samples, (D) Relative error between train and validation curvatures exhibits a local maximum, leading to an early stopping criterion, (E) Memorization cannot be detected just based on validation loss, (F) Memorization rises as models train further.

Proposition 4.2. *Let ϕ, ϕ' be parameter distributions as defined in 4.1 and m, M be defined as in (17) and (19). Fix $x_i \in S$. Then:*

$$-\mathbb{E}_{\phi'}[Gap_{\phi'}(x_i)] - M(x_i, S) \leq m(x_i, S) \leq \tag{20}$$

$$M(x_i, S) + \mathbb{E}_{\phi}[Gap_{\phi}(x_i)]. \tag{21}$$

We now bound M purely in terms of the properties of the leave-one-out model $h_{\phi'}$. We establish a number of mathematical preliminaries.

4.3 Mathematical Preliminaries

For a pair $z_i \in S$, $z_j \in S^{\setminus i}$ we use four scalar quantities throughout:

$$d_{ij} := \|z_i - z_j\|_2, \gamma_j := \|\nabla_x \ell(\phi, z_j)\|_2, \tag{22}$$

$$\kappa_i := \mathbb{E}_{\phi' \sim \mathcal{A}(S^{\setminus i})}[\|\nabla_z^2 \ell(\phi', z_i)\|_F^2] \tag{23}$$

$$C_{ij} := \left|\mathbb{E}_{\phi \sim \mathcal{A}(S)} \ell(\phi, z_i) - \mathbb{E}_{\phi' \sim \mathcal{A}(S^{\setminus i})} \ell(\phi', z_j)\right|. \tag{24}$$

Assumption 1. *(ℓ has ρ-Lipschitz hessians) The Hessian of the loss ℓ is Lipschitz in the z-variable, i.e for every $\phi \in \Phi$, there exists ρ (possibly dependent on ϕ) such that:*

$$\|\nabla_z^2 \ell(\phi, z') - \nabla_z^2 \ell(\phi, z)\|_2 \leq \rho \|z' - z\|_2, \tag{25}$$

for every $z, z' \in \mathcal{S}$.

Assumption 2. *(Loss behaves like MSE) We assume that the cost function $c(x, y)$ satisfies*

$$\|\nabla_y c(x, y)\|_2^2 \leq 2\beta c(x, y). \tag{26}$$

Note that any c that is β-Lipschitz in the y-variable satisfies (26) [SST10]. For instance, the MSE $c(x, y) = \|x - y\|_2^2$, or $c(x, y) = \|h(x) - y\|_2^2$ where $h(x)$ is itself Lipschitz satisfy (26).

Assumption 3. *(The underlying function is neural network-like) We assume that all $h^\phi = g(\phi_1 x, \phi_2)$ where $\phi = (\phi_1, \phi_2)$. Most neural network architectures satisfy this assumption.*

Note that the term in (28) is *not quite* $M(z_i, S)$ because the second term in the absolute value involves z_j (which is part of $S^{\setminus i}$) and not z_i.

Assumption 4. *We assume that $\mathcal{A}$ is a first order optimizer in the parameters, i.e there exist $\varepsilon_{\text{grad}}$ and $\varepsilon_{\text{loss}}$ such that*

$$\mathbb{E}_{\phi\sim\mathcal{A}(S)}[\|\nabla_\phi \ell(\phi, z_j)\|_2] \le \varepsilon_{\text{grad}}, \mathbb{E}_{\phi\sim\mathcal{A}(S)}[l(\phi, z_j)] \le \varepsilon_{\text{loss}}, \tag{27}$$

for every $z_j \in S$. Note that condition (27) is typically met by most deterministic and stochastic optimizers like SGD, Adam, or even full batch gradient descent.

Assumption 5. *We assume that under $\mathcal{A}(S)$, the weights ϕ are almost surely bounded by B.*

Assumption 6. *There exists a constant C independent of x_i, x_j in the leading order (i.e $C = O(1)$ in x_i, x_j) such that with high probability in p_Z^n, for any $z_i \in S, z_j \in S^{\setminus i}$ the following difference is uniformly bounded by C:*

$$|\mathbb{E}_{\phi\sim\mathcal{A}(S)}[\ell(h_S^\phi, z_i)] - \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\ell(h_{S^{\setminus i}}^{\phi'}, z_j)]| \le C. \tag{28}$$

Assumption 4 models $\mathcal{A}(S)$ as a first order optimizer that leads to bounded weights upon optimization. In the context of neural networks the weight-boundedness under standard optimization algorithms has been observed in several contexts, from deep linear models [SMG19], feedforward ReLU (or leaky ReLU) networks [DHL18] to transformers [AMSP24].

4.4 Training-Time Memorization is Controlled by Leave-One-Out Curvature

Theorem 4.3. *Under Assumption 1, for every $z_i \in S$ and every $z_j \in S^{\setminus i}$,*

$$|M(z_i; S)| \;\le\; C_{ij} + \gamma_j\, d_{ij} + \tfrac{3}{2}\,\kappa_i\, d_{ij}^2 + \tfrac{2\rho}{3}\, d_{ij}^3. \tag{29}$$

If Assumptions 2–5 hold, then

$$\gamma_j \;\le\; \frac{B\,\varepsilon_{\text{grad}}}{\|z_j\|_2} + \varepsilon_{\text{loss}}, \tag{30}$$

and if Assumption 6 holds as well, then with $d_{\text{NN}}(i) := \min_{j\neq i} d_{ij}$,

$$\begin{aligned} |M(z_i; S)| \;&\le C + d_{\text{NN}}(i)^2 \left(\tfrac{3}{2}\,\kappa_i \;+ \tfrac{2\rho}{3}\, d_{\text{NN}}(i)\right) \\ &\quad + \left(\frac{B\varepsilon_{\text{grad}}}{\min_j \|z_j\|} + \varepsilon_{\text{loss}}\right) d_{\text{NN}}(i). \end{aligned} \tag{31}$$

Corollary 4.4. *Let Assumptions 1–6 hold and additionally suppose $\varepsilon_{\text{grad}}, \varepsilon_{\text{grad}} \ll d_{\text{NN}}(i)$. Set $C(i) := (C + \mathbb{E}_{\phi'}[Gap_{\phi'}(x_i)])$. Then:*

$$-\,(C(i) + o(1)) - d_{\text{NN}}(i)^2 \left(\tfrac{3}{2}\,\kappa_i \;+ \tfrac{2\rho}{3}\, d_{\text{NN}}(i)\right) \tag{32}$$

$$\le m(x_i, S) \le d_{\text{NN}}(i)^2 \left(\tfrac{3}{2}\,\kappa_i \;+ \tfrac{2\rho}{3}\, d_{\text{NN}}(i)\right) + C(i) + o(1). \tag{33}$$

The proofs are provided in Appendix A. We make the following comments regarding Theorem 4.3:

1. Under Assumption 1 alone, (29) is a cubic expansion about an arbitrary second training point z_j. Each coefficient can be large and can depend on the query point z_i: the training-loss gap C_{ij}, the leave-one-out input gradient γ_j, the leave-one-out curvature κ_i, and the Hessian Lipschitz constant ρ.
2. Assumptions 2–5 put a neural network and a first-order method into the bound. When the algorithm overfits, $\varepsilon_{\text{grad}}$ and $\varepsilon_{\text{loss}}$ can be considered to be comparably small against $\kappa_i\, d_{ij}$, the curvature of the *leave-one-out* loss $\ell(\phi', z_i)$. Curvature thus bounds counterfactual memorization at the leading order, during overfitting.
3. Assumption 6 asks for a uniform C that is $O(1)$ in x. We can undertake a splitting argument to show that this $C = O(n)$–this is comparable to an analogous bound obtained in [RSR24] (Theorem 5.1) which is also $O(n)$.
4. [RSHR24] obtain $C = O(n)$ using β-uniform stability and remove the first order terms γ_i by comparing $\mathcal{A}(S)$ with $\mathcal{A}(S')$ on υ-adjacent datasets, where any two points lie at most a distance of υ apart, for small υ. [RSR24] remark that a training set must be augmented *post-hoc* to make it υ-adjacent. Here Assumption 6 replaces the uniform stability assumption. Moreover, we do not require such a condition on the data–instead we restrict the model class to be structured like a neural network in order to use its inductive biases to make the first order terms negligible. Overall, our bound does not require uniform stability, υ-adjacency, or bounded loss as is required by Theorem 5.1 in [RSR24]–however, we make more structural but nonetheless realistic assumptions on the model class and the algorithms at hand.

4.5 Application to Diffusion Models

We can make further specifications with additional assumptions for the denoising loss (7) and the fact that typically, the hypothesis class h^ϕ contains linear functions.

Assumption 7. *We assume that for any $M \in \mathbb{R}^{d\times m}$ and $b \in \mathbb{R}^m$ there exists some ϕ such that $h^\phi = Mx + b$.*

In objective (7) we train a denoiser $D_\phi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$. Thus we require that the underlying function class contain at least all linear denoisers. This assumption is satisfied by ReLU networks or ReLU U-Nets. Next we state the PL-inequality assumption used for simplifying the analysis of first-order methods in non-convex optimization. Note that neural network loss landscapes have been known to satisfy the local-PL inequality below for the parameter regions of most commonly used neural networks [LZB22].

Assumption 8. *Let ϕ be the parameters and $\nabla_\phi \ell(\phi, S)$ the parameter gradient of the empirical loss. We assume that there exists $\mu > 0$ such that almost surely under $\mathcal{A}(S)$,*

$$(\ell(\phi, S) - \ell(S^*)) \leq \frac{1}{2\mu}\|\nabla_\phi \ell(\phi, S)\|_2^2 \tag{34}$$

Proposition 4.5. *Under assumptions 1–8, if ℓ is given by (7), there exists $A(T)$ depending on f, g, T such that the constant C in assumption 6 is of the form:*

$$C \leq \frac{n}{\mu}\varepsilon_{\text{grad}} + nA(T). \tag{35}$$

We provide a proof in Appendix A. Note that in this setting the constant C is comparable to the constant c_1 given in Eq. 8 in [RSR24].

4.6 Held-Out Curvature During Training

Finally, we validate our bound on the DDPMs on Imagenet, to check whether held-out curvature can stand as a proxy for counterfactual memorization. In order to track it during training, we assume bounded ELBO gaps in (10) and track the counterfactual loss difference (19) instead of the log odds (17). Since training a diffusion model on $S^{\setminus i}$ for every i is prohibitively expensive, we remove the top 10 most memorized examples for the DDPMs corresponding to $n = 128$ and $n = 1024$. We then estimate $M(x_i, S)$ for these ten points, with the loss on the $S' := S \setminus \{\text{top 10 memorized points}\}$ with parameters ϕ' evaluated on x_i as an estimator for $\ell(h^{\phi'}_{S^{\setminus i}}, x_i)$. We present our results in Figures 3 and 4 where we track:

$$\Delta\ell(\tau) := \ell(h^\phi_S, z_i) - \ell(h^{\phi'}_{S'}, z_i). \tag{36}$$

Note that $\Delta\ell$ is the signed version of (19). We observe the following phenomenon with $\Delta\ell$: as training progresses, $\Delta\ell$ drops sharply at the initial phase of training, showing that the full model attempts to fit at the seen example x_i. Then the loss stabilizes as both models enter their respective memorization phases. As the f_{mem} approaches 100% $\Delta(\tau)$ starts dropping quickly again. Throughout this whole process, however, the curvature of the held out models at the held out samples roughly corresponds to the drops and stability of $\Delta(\tau)$, thus suggesting that our bound (29) is a meaningful proxy for the counterfactual loss. At the very tail of training, the curvature for the held-out models *rises*, reflecting the local minimum phenomenon seen in Figures 1 and 2. However, $\Delta(\tau)$ keeps falling, suggested that the held out point's influence on $\Delta(\tau)$ via curvature breaks down in the highly overfitted regime.

5 Conclusion

In this paper, we introduced the input-loss curvature as a sampling-free proxy for f_{mem}. Averaged over the training set, it tracks f_{mem} on ImageNet and on CelebA as training proceeds, albeit it is not sufficient to detect the onset or prevalence of memorization. Instead, we find that the step at which the validation curvature separates from the training curvature marks the transition into a memorizing model. However, over individual points, the input loss curvature cannot reflect the copy count of individual samples. For that question, we use a counterfactual definition, the change in loss and in log-likelihood at x when x is removed from the training set, and we prove that this quantity is influenced, to leading order in the displacement of x, by the input-loss curvature of the leave-one-out model.

We enumerate the limitations of our findings below:

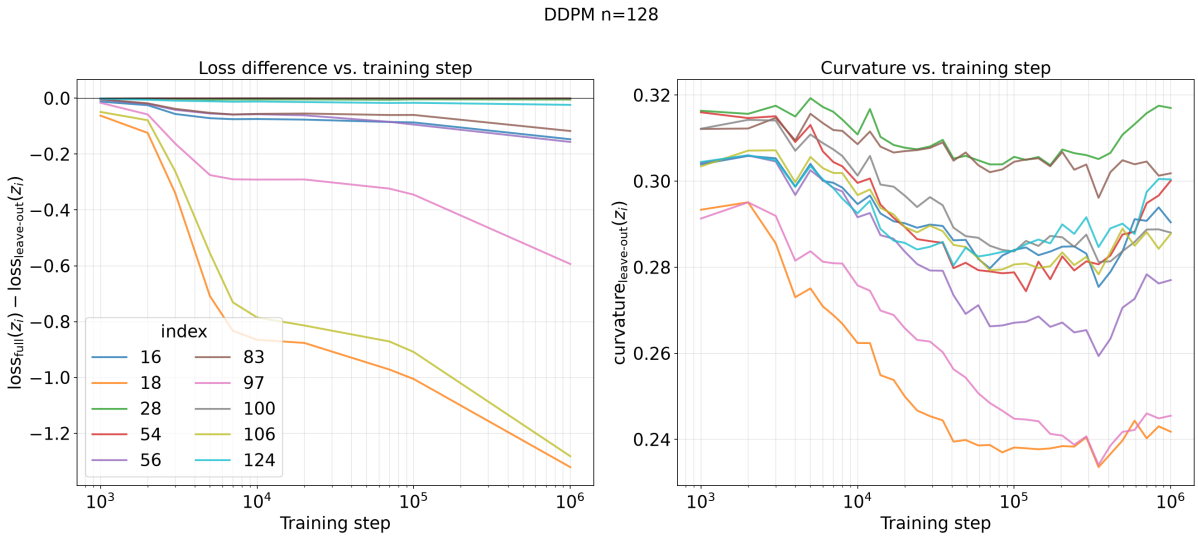


Figure 3: **Loss difference and curvature for Imagenet (n=128) DDPM noise prediction - leave-10-out training**: (left) difference between per sample loss of full model and leave-10-out model vs. training step for each of the 10 left-out training samples, (right) per sample curvature of the leave-10-out model vs. training step for each of the 10 left-out training samples.

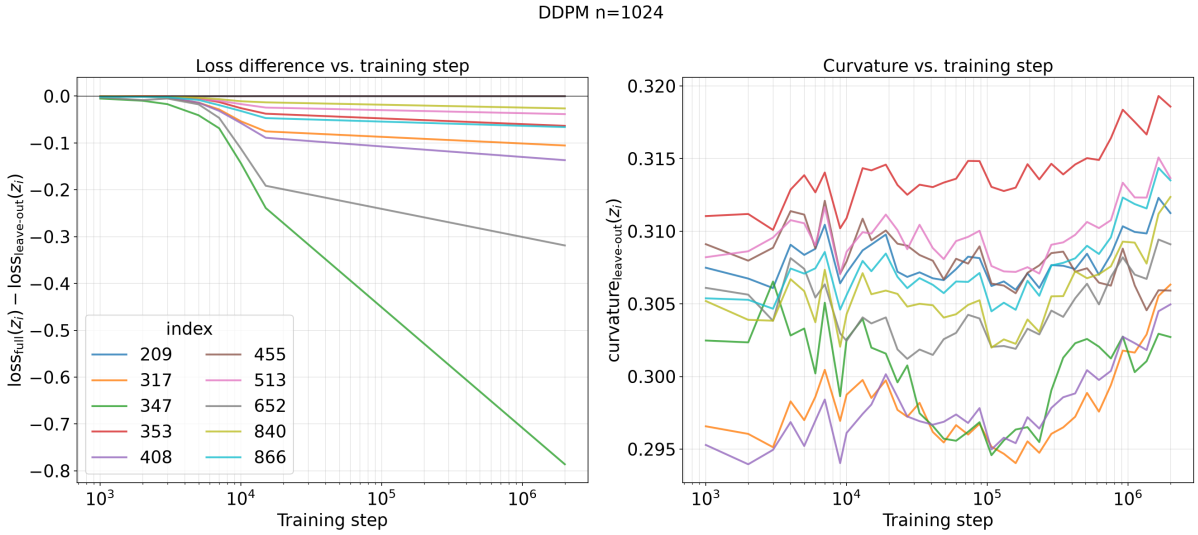


Figure 4: **Loss difference and curvature for Imagenet (n=1024) DDPM noise prediction - leave-10-out-training**: (left) difference between per sample loss of full model and leave-10-out model vs. training step for each of the 10 left-out training samples, (right) per sample curvature of the leave-10-out model vs. training step for each of the 10 left-out training samples.

1. We provide narrow evidence ranging over only two datasets, ImageNet and CelebA. Moreover, a comparison between the various training losses used to train diffusion models merits further investigation that is not performed at length here.

2. We propose using the local maximum of the relative train–validation curvature error as a stopping criterion. However, our experiments show that this local maximum indicates a period of already high memorization. An improved statistic would be needed to locate the onset of memorization. For small datasets, Figure 2 shows that the norm of the loss gradient averaged over the validation data might prove useful, but its role and behaviour for larger datasets is not well-understood.

3. The bound is a plausibility argument and depends on a list of assumptions on the loss, the model, and the optimizer. It is also unsettled whether the curvature term dominates the ELBO gap at x, which is the comparison that decides whether curvature is the leading contribution to (17).

Acknowledgment

This work was supported in part by NSF DMS 2502561.

References

[AAD+24] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[AMSP24] Murdock Aubry, Haoming Meng, Anton Sugolov, and Vardan Papyan. Transformer block coupling and its correlation with generalization in LLMs. *arXiv preprint arXiv:2407.07810*, 2024.

[And82] Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[AVS+24] Beatrice Achilli, Enrico Ventura, Gianluigi Silvestri, Bao Pham, Gabriel Raya, Dmitry Krotov, Carlo Lucibello, and Luca Ambrogioni. Losing dimensions: Geometric memorization in generative diffusion. *arXiv preprint arXiv:2410.08727*, 2024.

[BBdBM24] Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15:9957, 2024.

[BdH15] Anton Bovier and Frank den Hollander. *Metastability: A Potential-Theoretic Approach*, volume 351 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2015.

[BDK+25] Ricardo Baptista, Agnimitra Dasgupta, Nikola B Kovachki, Assad Oberai, and Andrew M Stuart. Memorization and regularization in generative diffusion models. *arXiv preprint arXiv:2501.15785*, 2025.

[Ber13] Nils Berglund. Kramers' law: Validity, derivations and generalisations. *Markov Processes and Related Fields*, 19(3):459–490, 2013.

[BM23] Giulio Biroli and Marc Mézard. Generative diffusion in very large dimensions. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(9):093402, 2023.

[BUBM25] Théo Bonnaire, Raphaël Urfin, Giulio Biroli, and Marc Mézard. Why diffusion models don't memorize: The role of implicit dynamical regularization in training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[CCN+22] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy*, pages 1897–1914, 2022.

[CHN+23] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium*, pages 5253–5270, 2023.

[CTW+21] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium*, pages 2633–2650, 2021.

[DDS+09] 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*, pages 248–255, 2009.

[DHL18] Simon S. Du, Wei Hu, and Jason D. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*, 31, 2018.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[Fel20] Vitaly Feldman. Does learning require memorization? A short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[FZ20] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

[GDP+23] Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

[GRR24] Isha Garg, Deepak Ravikumar, and Kaushik Roy. Memorization through the lens of curvature of loss function around samples. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[HLC21] Chin-Wei Huang, Jae Hyun Lim, and Aaron C. Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34:22863–22876, 2021.

[KGSM24] Zahra Kadkhodaie, Florentin Guth, Eero P. Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *International Conference on Learning Representations*, 2024.

[Kin14] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LCBH+23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[LGL23] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

[LLWT15] 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.

[LZB22] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[MY21] Chao Ma and Lexing Ying. On linear stability of SGD and input-smoothness of neural networks. *Advances in Neural Information Processing Systems*, 34:16805–16817, 2021.

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8162–8171, 2021.

[NP06] Yurii Nesterov and Boris T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[RBL+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.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer International Publishing, 2015.

[RPG+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[RSHR24] Deepak Ravikumar, Efstathia Soufleri, Abolfazl Hashemi, and Kaushik Roy. Unveiling privacy, memorization, and input curvature links. In *Proceedings of the 41st International Conference on Machine Learning*, pages 42192–42212, 2024.

[RSR24] Deepak Ravikumar, Efstathia Soufleri, and Kaushik Roy. Curvature clues: Decoding deep learning privacy with input loss curvature. *Advances in Neural Information Processing Systems*, 37, 2024.

[SDWMG15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, 2015.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[SME21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[SMG19] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[SSDK+21] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[SSG+23a] 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.

[SSG+23b] 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.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pages 3–18, 2017.

[SST10] Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Optimistic rates for learning with a smooth loss. *arXiv preprint arXiv:1009.3896*, 2010.

[Vin11] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[WDMG25] Dana Weitzner, Mauricio Delbracio, Peyman Milanfar, and Raja Giryes. The diffusion process as a correlation machine: Linear denoising insights. *Transactions on Machine Learning Research*, 2025.

[YCKR23] TaeHo Yoon, Joo Young Choi, Sehyun Kwon, and Ernest K. Ryu. Diffusion probabilistic models generalize when they fail to memorize. In *ICML 2023 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2023.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[ZIL+23] Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362, 2023.

A Proofs for Section 4

Lemma A.1 (Lemma 1 from [NP06]). *Suppose $\ell(\phi, z)$ satsifies Assumption 1 of ρ-Lipschitz Hessians. Then for any $z, z' \in \mathbb{R}^d$*

$$-\frac{\rho}{6}\|z' - z\|_2^3 \leq \ell(\phi, z') - \ell(\phi, z) - \langle \nabla \ell(\phi, z), z - z' \rangle - (z' - z)^\top \nabla_z^2 \ell(\phi, z)(z' - z) \leq \frac{\rho}{6}\|z' - z\|_2^3. \tag{37}$$

Using Assumption (1) we can also obtain the following bound on the gradients of ℓ:

Lemma A.2. *Suppose $\ell(\phi, z)$ satisfies Assumption 1 of ρ-Lipschitz Hessians. Then for any $z, z' \in \mathbb{R}^d$*

$$\|\nabla_z \ell(\phi, z') - \nabla_z \ell(\phi, z)\|_2 \leq \|\nabla_z^2 \ell(\phi, z)\|_{2\to 2}\|z' - z\|_2 + \frac{\rho}{2}\|z' - z\|_2^2. \tag{38}$$

Here $\|\nabla_z^2 \ell(\phi, z)\|_{2\to 2}$ is the operator norm of the Hessian.

Following an argument of Ma & Ying [MY21] we use the multiplicative nature of neural networks to prove a bound on the input gradients in terms of the parameter gradients.

Lemma A.3. *Suppose $\ell(\phi, z) = g(\phi_1 z, \phi_2, z)$ and there exists a constant C_2 such that $\|\nabla_c g\| \leq C_2 g$. Then we have*

$$\|\nabla_z \ell(\phi, z)\|_2 \leq \frac{\|\phi_1\|}{\|z\|}\|\nabla_{\phi_1} \ell(\phi, z)\| + C_2 \ell(\phi, z). \tag{39}$$

Proof. We compute the gradient with respect to the inputs:

$$\nabla_z \ell(\phi, z) = \nabla_z g(\phi_1 z, \phi_2, z) = \phi_1^\top \nabla_a g(\phi_1 z, \phi_2, z) + \nabla_c g(\phi_1 z, \phi_2, z), \tag{40}$$

$$\nabla_{\phi_1} \ell(\phi, z) = \nabla_a g(\phi_1 z, \phi_2, z) z^\top. \tag{41}$$

Multiplying the first equation on the right by $z^\top$ we get

$$\nabla_z \ell(\phi, z) z^\top = \phi_1^\top \nabla_a g(\phi_1 z, \phi_2, z) z^\top + \nabla_c g(\phi_1 z, \phi_2, z) z^\top \tag{42}$$

$$\iff \nabla_z \ell(\phi, z) z^\top = \phi_1^\top \nabla_{\phi_1} \ell(\phi, z) + \nabla_c g(\phi_1 z, \phi_2, z) z^\top \tag{43}$$

$$\iff \nabla_z \ell(\phi, z) = \frac{\phi_1^\top \nabla_{\phi_1} \ell(\phi, z) z}{\|z\|_2^2} + \nabla_c g(\phi_1 z, \phi_2, z). \tag{44}$$

Now, taking norms and using our assumption we have:

$$\|\nabla_z \ell(\phi, z)\|_2 \leq \frac{\|\phi_1\|}{\|z\|}\|\nabla_{\phi_1} \ell(\phi, z)\| + C_2 \ell(\phi, z). \tag{45}$$

□

Before going further, we remark on the significance of Lemma A.3. In machine learning, we optimize the loss with respect to the parameters in the model ϕ. However, since we are interested in the effect of the inclusion of a single training sample, we are more interested in the behaviour of the loss with respect to its *inputs* z. Lemma A.3 gives us a way to relate these two, specifically relating the sizes of the input and parameter gradients of the loss. In particular, it says that the input gradients can be (roughly) controlled by the parameter gradients, which we explicitly control via optimization.

Using the cubic envelope from Lemma (A.1) we may bound the memorization term $M(z_i, S)$ from (46).

Lemma A.4. *Let S be a finite dataset and let $S' \subseteq S$ and let $z_i \in S \setminus S'$. Then under assumptions (1) and (6), for any $z_j \neq z_i \in S'$ we have:*

$$\begin{aligned}\mathbb{E}_\phi[\ell(h_S^\phi, z_i)] - \mathbb{E}_{\phi'}[\ell(h_{S'}^{\phi'}, z_j)] \leq C &+ |\langle \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z \ell(h_{S'}^{\phi'}, z_i)], z_j - z_i \rangle| \\ &+ |(z_j - z_i)^\top \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z^2 \ell(h_{S'}^{\phi'}, z_i)](z_j - z_i)| + \frac{\rho}{6}\|z_j - z_i\|_2^3.\end{aligned} \tag{46}$$

Proof. Let $\phi' \sim \mathcal{A}(S')$. For z_j, z_i, we have, using Lemma A.1 that:

$$-\ell(h^{\phi'}_{S'}, z_j) \geq -\ell(h^{\phi'}_{S'}, z_i) - \langle \nabla_z \ell(h^{\phi'}_{S'}, z_i), z_j - z_i \rangle - (z_j - z_i)^\top \nabla_z^2 \ell(h^{\phi'}_{S'}, z_i)(z_j - z_i) - \frac{\rho}{6}\|z_j - z_i\|_2^3, \tag{47}$$

$$\ell(h^{\phi'}_{S'}, z_j) \leq \ell(h^{\phi'}_{S'}, z_i) + \langle \nabla_z \ell(h^{\phi'}_{S'}, z_i), z_j - z_i \rangle + (z_j - z_i)^\top \nabla_z^2 \ell(h^{\phi'}_{S'}, z_i)(z_j - z_i) + \frac{\rho}{6}\|z_j - z_i\|_2^3. \tag{48}$$

Using the monotonicity of expectation we therefore get:

$$-\mathbb{E}_{\phi\sim\mathcal{A}(S')}[\ell(h^{\phi'}_{S'}, z_j)] \geq -\mathbb{E}_{\phi\sim\mathcal{A}(S')}[\ell(h^{\phi'}_{S'}, z_i)] - \langle \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z \ell(h^{\phi'}_{S'}, z_i)], z_j - z_i \rangle \tag{49}$$

$$- (z_j - z_i)^\top \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z^2 \ell(h^{\phi'}_{S'}, z_i)](z_j - z_i) - \frac{\rho}{6}\|z_j - z_i\|_2^3, \tag{50}$$

$$\mathbb{E}_{\phi\sim\mathcal{A}(S')}[\ell(h^{\phi'}_{S'}, z_j)] \leq \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\ell(h^{\phi'}_{S'}, z_i)] + \langle \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z \ell(h^{\phi'}_{S'}, z_i)], z_j - z_i \rangle \tag{51}$$

$$+ (z_j - z_i)^\top \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z^2 \ell(h^{\phi'}_{S'}, z_i)](z_j - z_i) + \frac{\rho}{6}\|z_j - z_i\|_2^3. \tag{52}$$

Plugging these into the two-sided bound implied by Assumption 6 we get:

$$C \geq \mathbb{E}_{\phi}[\ell(h^{\phi}_{S}, z_i)] - \mathbb{E}_{\phi'}[\ell(h^{\phi'}_{S'}, z_j)] \tag{53}$$

$$\geq \mathbb{E}_{\phi}[\ell(h^{\phi}_{S}, z_i)] - \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\ell(h^{\phi'}_{S'}, z_i)] - \langle \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z \ell(h^{\phi'}_{S'}, z_i)], z_j - z_i \rangle \tag{54}$$

$$- (z_j - z_i)^\top \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z^2 \ell(h^{\phi'}_{S'}, z_i)](z_j - z_i) - \frac{\rho}{6}\|z_j - z_i\|_2^3. \tag{55}$$

By rearranging, we have

$$\mathbb{E}_{\phi}[\ell(h^{\phi}_{S}, z_i)] - \mathbb{E}_{\phi'}[\ell(h^{\phi'}_{S'}, z_j)] \leq C + \langle \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z \ell(h^{\phi'}_{S'}, z_i)], z_j - z_i \rangle \tag{56}$$

$$+ (z_j - z_i)^\top \mathbb{E}_{\phi\sim\mathcal{A}(S')}[\nabla_z^2 \ell(h^{\phi'}_{S'}, z_i)](z_j - z_i) + \frac{\rho}{6}\|z_j - z_i\|_2^3. \tag{57}$$

Using the same tactic, we may prove a lower bound for $\mathbb{E}_{\phi}[\ell(h^{\phi}_{S}, z_i)] - \mathbb{E}_{\phi'}[\ell(h^{\phi'}_{S'}, z_j)]$ using the negative of the term above. This yields a bound for the absolute value, and recalling the definition of M yields the bound (46). □

We can now and prove our main result 4.3.

Proof. First, we construct the high probability event under which the bound should hold. Set the threshold in assumption 4 to be $o(n^{-1/2})$ where n is the size of the dataset. Then fix a threshold $1 - \delta(n)$ such that both events under which assumptions 6 and 5 hold. In this high probability event we apply Lemma A.4 to the set $S' = S^{\setminus i} := S \setminus \{z_i\}$ to obtain (46):

$$M(z_i, S) \leq C + \underbrace{|\langle \mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)], z_j - z_i \rangle|}_{A}$$

$$+ \underbrace{|(z_j - z_i)^\top \mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\nabla_z^2 \ell(h^{\phi'}_{S^{\setminus i}}, z_i)](z_j - z_i)| + \frac{\rho}{6}\|z_j - z_i\|_2^3}_{B}. \tag{46}$$

To bound (A), we use Cauchy-Schwarz and Jensen's inequality:

$$|\langle \mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)], z_j - z_i \rangle| \leq \mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\|\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_2]\|z_j - z_i\|_2. \tag{58}$$

We now apply Lemma A.2 on $\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_2$ and take expectations to obtain:

$$\mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\|\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_2]\|z_j - z_i\|_2 \leq \mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\|\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_j)\|_2]\|z_j - z_i\|_2 \tag{59}$$

$$+ \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\|\nabla_z^2 \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_{2\to 2}]\|z_j - z_i\|_2^2 + \frac{\rho}{2}\|z_j - z_i\|_2^3. \tag{60}$$

We now apply Lemma A.3 to (59) and thus get:

$$\mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\|\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_j)\|_2]\|z_j - z_i\|_2 \leq \left(\mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}\left[\frac{\|\phi'_1\|}{\|z\|}\|\nabla_{\phi'_1} \ell(h^{\phi'}_{S^{\setminus i}}, z_j)\| + 2\beta\ell(h^{\phi'}, z_j)\right]\right)\|z_j - z_i\|_2 \tag{61}$$

$$= \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}\left(\frac{B}{\min_k \|z_k\|}\varepsilon_{\text{grad}} + 2\beta\varepsilon_{\text{loss}}\right)\|z_j - z_i\|_2 \tag{62}$$

The last equality follows due to assumption 4 and 5 which results in both the loss and the gradients of the loss of a trained model at a training sample to be small (compared to the parameters and the nearest neighbor distance $\|z_j - z_i\|$). Putting (62) with (59) we get:

$$|\langle \mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\nabla_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)], z_j - z_i\rangle| \le \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\|\nabla^2_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_{2\to 2}]\|z_j - z_i\|_2^2 \tag{63}$$

$$+ \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}\left(\frac{B}{\min_k \|z_k\|}\varepsilon_{\text{grad}} + 2\beta\varepsilon_{\text{loss}}\right)\|z_j - z_i\|_2 + O(\|z_j - z_i\|_2^3). \tag{64}$$

To bound (B) we use an operator norm upper bound along with Jensen's inequality to bound the quadratic term in (46) also by the operator norm of the hessian appearing in (64):

$$|(z_j - z_i)^\top \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\nabla^2_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)](z_j - z_i)| = |(z_j - z_i)^\top \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\nabla^2_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)(z_j - z_i)]| \tag{65}$$

$$\le \|\mathbb{E}_{\phi\sim\mathcal{A}(S^{\setminus i})}[\nabla^2_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)(z_j - z_i)]\|\|z_j - z_i\| \tag{66}$$

$$\le \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\|\nabla^2_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_{2\to 2}]\|z_j - z_i\|_2^2 \tag{67}$$

$$\le \mathbb{E}_{\phi'\sim\mathcal{A}(S^{\setminus i})}[\|\nabla^2_z \ell(h^{\phi'}_{S^{\setminus i}}, z_i)\|_F^2]\|z_j - z_i\|_2^2 \tag{68}$$

Plugging (64) and (68) yields (29). □

A.1 Proof of Proposition 4.5

For brevity we set $z = z_i, z' := z_j$, and $S' = S^{\setminus i}$. First, since the function class h^ϕ contains linear denoisers, the performance of h^ϕ under $\mathcal{A}(S)$ is *better* than any linear denoiser. However, the best linear denoiser attains an MSE dependent on T, f, g [WDMG25] that we term $A(T)$. Thus $\ell^*(S) \le A(T)$. We may now decompose the difference in assumption 6 as follows by introducing the rest of the dataset S, S':

$$\begin{aligned}
\ell(h^\phi, z) - \ell(h^{\phi'}, z') &= |S|\left(\ell(h^\phi, S) - \ell^*(S)\right) - |S'|\left(\ell(h^{\phi'}, S') - \ell^*(S')\right) + |S'|\left(\ell^*(S) - \ell^*(S')\right) \\
&\quad + |S \setminus S'|\ell^*(S) + \sum_{z_k\in S', z_k\neq z'} \ell(h^{\phi'}, z_k) - \sum_{z_k\in S, z_k\neq z} \ell(h^\phi, z_k) \\
&\le |S|\left(\ell(h^\phi, S) - \ell^*(S)\right) - |S'|\left(\ell(h^{\phi'}, S') - \ell^*(S')\right) + |S'|(\ell^*(S) - \ell^*(S')) \\
&\quad + |S - S'|\ell^*(S).
\end{aligned}$$

Here the inequality on the third line follows because of the following calculation:

$$\begin{aligned}
&\sum_{z_k\in S', z_k\neq z'} \ell(h^{\phi'}, z_k) - \sum_{z_k\in S, z_k\neq z} \ell(h^\phi, z_k) \\
&\le \sum_{z_k\in S'} \ell(h^{\phi'}, z_k) - \sum_{z_k\in S'} \ell(h^\phi, z_k) = |S'|(\ell(h^{\phi'}, S') - \ell(h^\phi, S')) \le 0.
\end{aligned}$$

Using the PL-inequality in assumption 8, the bound $\ell^*(S) \le A(T)$ and the fact that $l^*(S') \ge 0$ we get:

$$\ell(h^\phi, z) - \ell(h^{\phi'}, z') \le \frac{n}{\mu}\varepsilon_{\text{grad}} + nA(T). \tag{69}$$

B Experimental Details

Here we describe our experimental details. Note that we also conducted analogous versions of our experiments for *flow-matching* which we describe below.

Flow matching [LCBH+23, LGL23] learns a time-dependent velocity field $v_\theta(x, t)$ whose flow transports a noise distribution p_0 to the data distribution p_1. Given $X_0 \sim p_0$ and $X_1 \sim p_1$, one forms the linear interpolant $X_t = (1-t)X_0 + tX_1$ for $t \in [0, 1]$, whose conditional velocity is $\frac{d}{dt}X_t = X_1 - X_0$. The field is trained by regressing onto this target,

$$\ell(\phi, x) = \mathbb{E}_{t,\, X_0,\, X_1}\left[\left\|v_\theta(X_t, t) - (X_1 - X_0)\right\|_2^2 \mid X_0 = x\right], \tag{70}$$

and the minimizer is the marginal velocity $v^\star(x,t) = \mathbb{E}[X_1 - X_0 \mid X_t = x]$. Samples are then generated by integrating the ODE $\dot{x}_t = v_\theta(x_t, t)$ from $t = 0$ to $t = 1$.

We conduct numerical experiments on the Imagenet and CelebA datasets. We use ImageNet-1k [DDS+09, RDS+15] and convert to RGB downscaled images of size $d = 64 \times 64$. For Imagenet, we train on $n \in \{128, 256, 512, 1024, 2048, 4096, 10240\}$. We convert the CelebA [LLWT15] face images to grayscale and downsample them to size $d = 32 \times 32$. We use a training set size of $n \in \{128, 10240\}$ for CelebA, yielding a memorizing vs. generalizing model. All our score models have a U-Net architecture [RFB15] with four resolution levels, base channel width $W = 64$, and channel multipliers 1, 2, 4, and 4 respectively. Self-attention is applied at the two lowest-resolution levels, and each level has two residual blocks in the encoder and three in the decoder. The models are conditioned on diffusion time t through the sinusoidal embedding [VSP+17] in every residual block, so a single network serves all noise levels. We train the network in one of three ways: DDPM [HJA20]with either noise prediction (ϵ-prediction) or clean-sample prediction (x_0-prediction), or flow matching [LCBH+23]. The DDPM noise prediction predicts the injected Gaussian noise at diffusion time t using the Adam optimizer [Kin14] with a fixed batch size of 128. DDPM sample prediction takes the noisy images x_t and time t and predicts the clean image x_0 directly using loss $\mathrm{MSE}(\hat{x_0}(x_t, t), x_0)$. DDPM models are sampled using DDIM with 100 steps and stochasticity parameter $\eta = 0$, starting from $x_T \sim \mathcal{N}(0, I)$. For the noise prediction models, the predicted noise $\hat{\epsilon}$ is converted to a clean image estimate $\hat{x_0} = (x_t - \sqrt{(1-\bar{\alpha}_t)}\hat{\epsilon})/\sqrt{\bar{\alpha}_t}$ before stepping to the next time. The sample prediction models use the same DDIM sampler with the network output taken as the clean image estimate $\hat{x_0}$ directly. The flow matching networks predict the velocity $u = x_0 - \epsilon$ along the linear path $x_t = (1-\tau)\epsilon + \tau x_0$, where $\epsilon \sim \mathcal{N}(0, I)$ is the noise, x_0 is the training image, and τ is the normalized time. Training minimizes $\mathrm{MSE}(\hat{v}(x_t, t), u)$, and sampling integrates $\frac{\mathrm{d}x}{\mathrm{d}\tau} = \hat{v}$ from $\tau = 0$ (noise) to $\tau = 1$ (data) with 100 Euler steps.

C Generated Samples

Figures 5-12 shows generated samples at the pre-memorization, beginning of memorization, and full memorization stages with their nearest training neighbor for DDPM noise prediction and flow matching on CelebA with $n \in \{128, 10240\}$ as well as the leave-10-out models for Imagenet with $n \in \{128, 1024\}$.

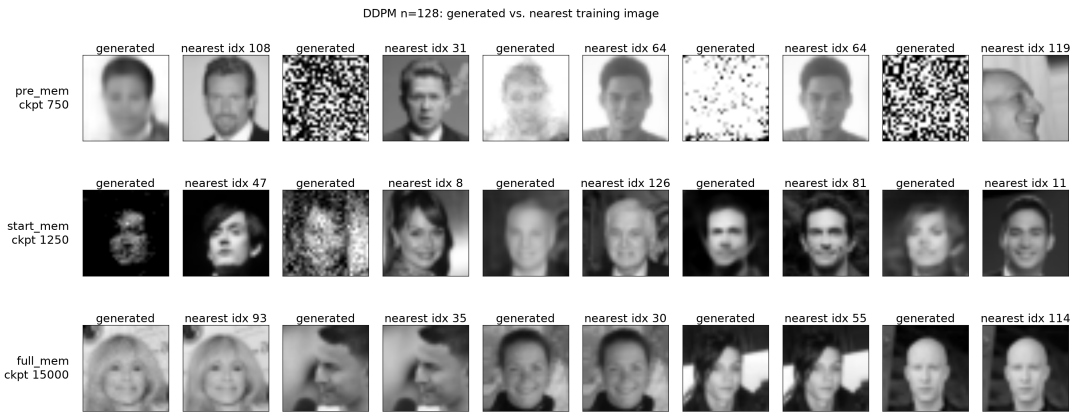


Figure 5: **Generated samples vs. nearest training image for CelebA (n=128) DDPM**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

Figures 13–20 show generated samples and their nearest training images for models trained on all n images: DDPM noise prediction with $n \in \{128, 256, 512, 1024, 2048, 4096\}$ and flow matching with $n \in \{128, 1024\}$. Each row contains five generated–training image pairs. All memorization fractions are measured over $10{,}000$ generations.

For models with sustained memorization, pre_mem denotes the last evaluated checkpoint with $f_{\mathrm{mem}} = 0$ before onset, and start_mem denotes the first checkpoint after which f_{mem} stays above zero. Full memorization (full_mem) denotes the first evaluated checkpoint with $f_{\mathrm{mem}} \geq 95\%$. For DDPM with $n = 512$ and $n = 1024$, which never reach this threshold, peak_mem denotes the highest observed memorization fraction. For DDPM with $n = 4096$, the two rows instead show a zero-copy checkpoint followed by an isolated copy event.

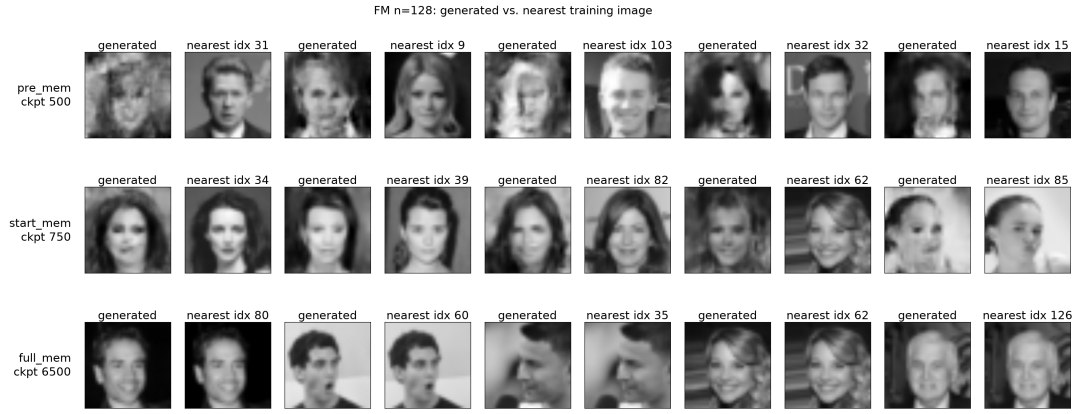


Figure 6: **Generated samples vs. nearest training image for CelebA (n=128) flow matching**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

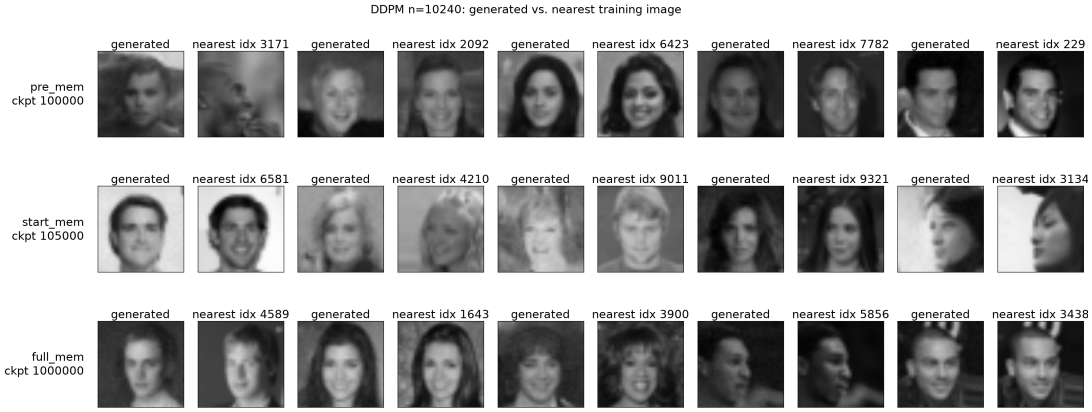


Figure 7: **Generated samples vs. nearest training image for CelebA (n=10240) DDPM**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

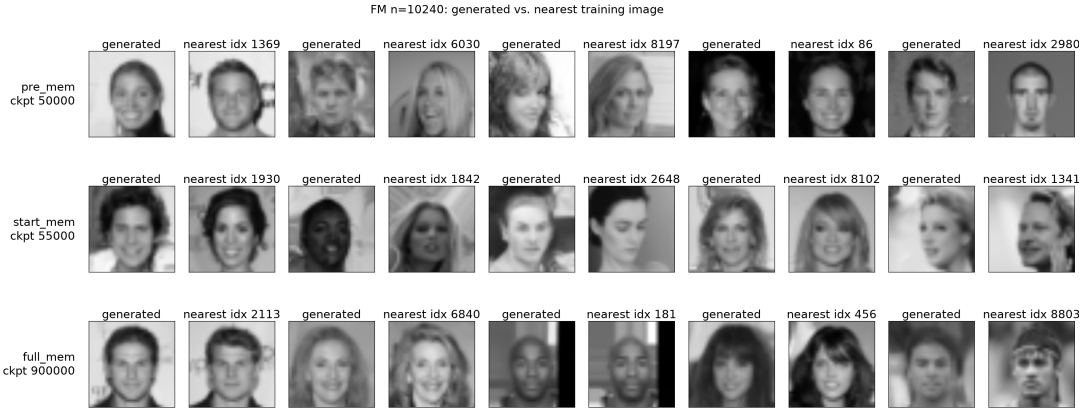


Figure 8: **Generated samples vs. nearest training image for CelebA (n=10240) flow matching**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

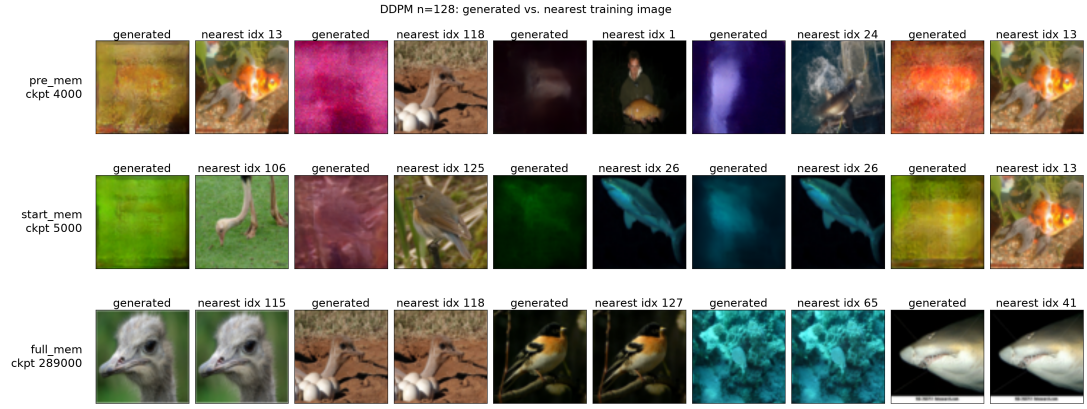


Figure 9: **Generated samples vs. nearest training image for Imagenet (n=128) DDPM leave-10-out model**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

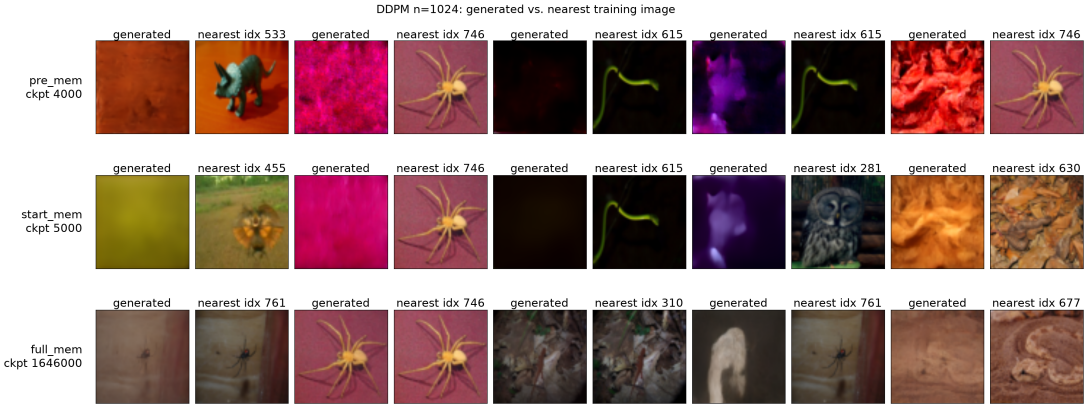


Figure 10: **Generated samples vs. nearest training image for Imagenet (n=1024) DDPM leave-10-out model**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

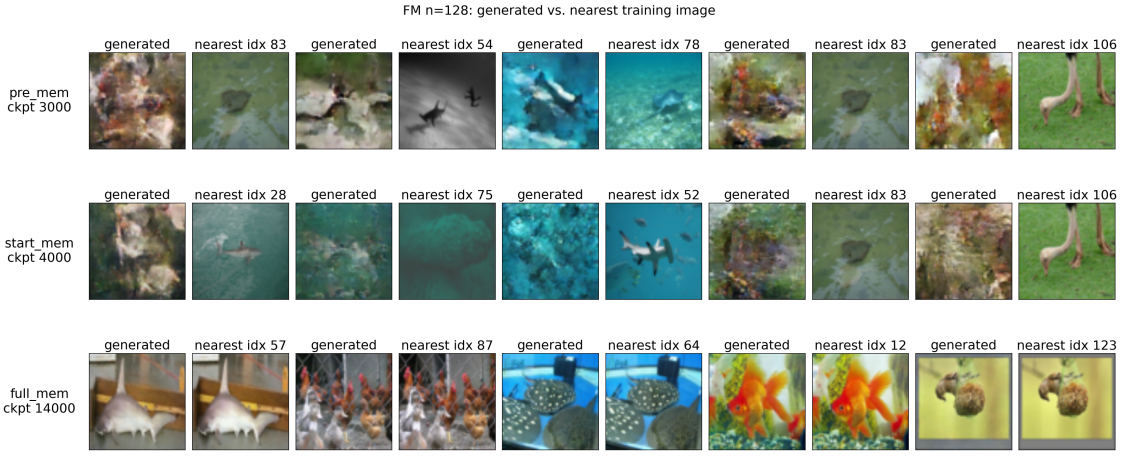


Figure 11: **Generated samples vs. nearest training image for Imagenet (n=128) flow matching leave-10-out model**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

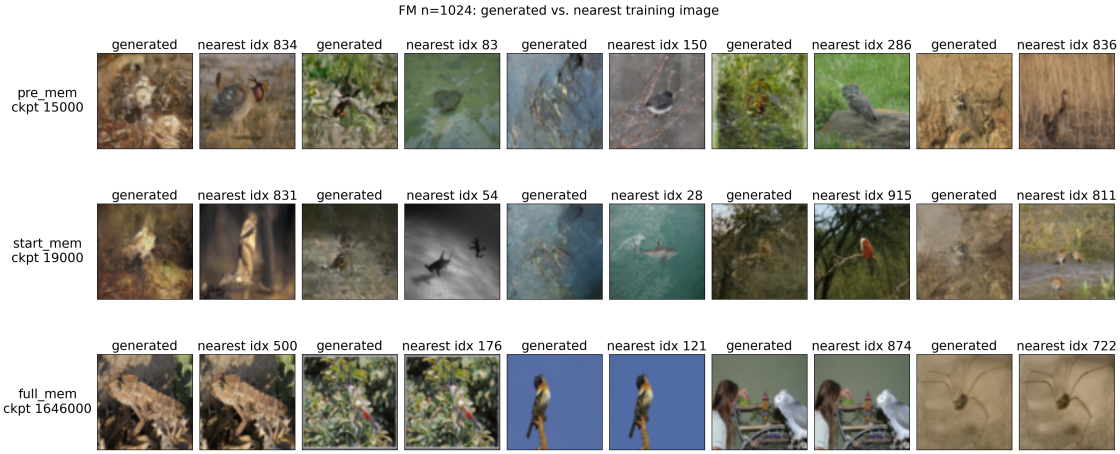


Figure 12: **Generated samples vs. nearest training image for Imagenet (n=1024) flow matching leave-10-out model**: generated samples at the pre-memorization, start of memorization, and full memorization stages alongside their nearest neighbor in the training set.

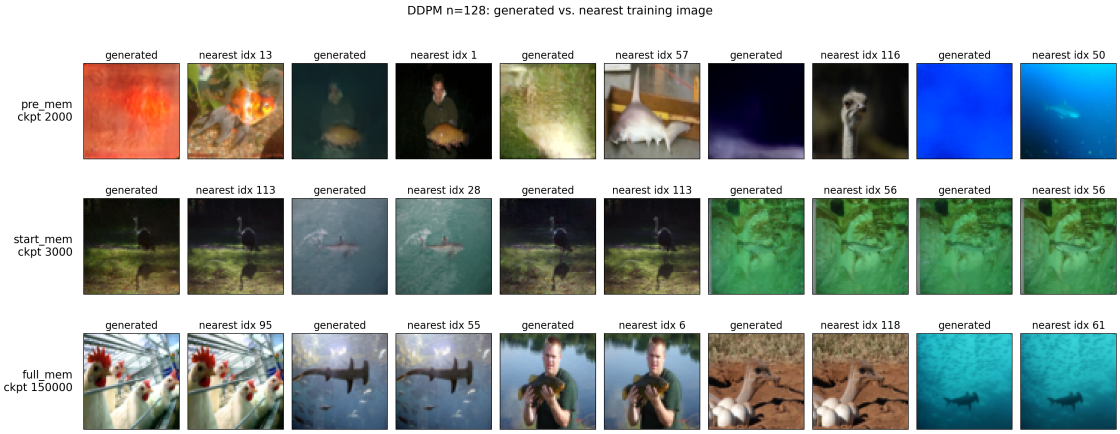


Figure 13: **Generated samples vs. nearest training image for ImageNet** ($n = 128$) **DDPM.** pre_mem: step 2,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 3,000, $f_{\text{mem}} = 0.10\%$ (10 copies of 10^4), five randomly selected copies. full_mem: step 150,000, $f_{\text{mem}} = 99.8\%$, five randomly selected copies.

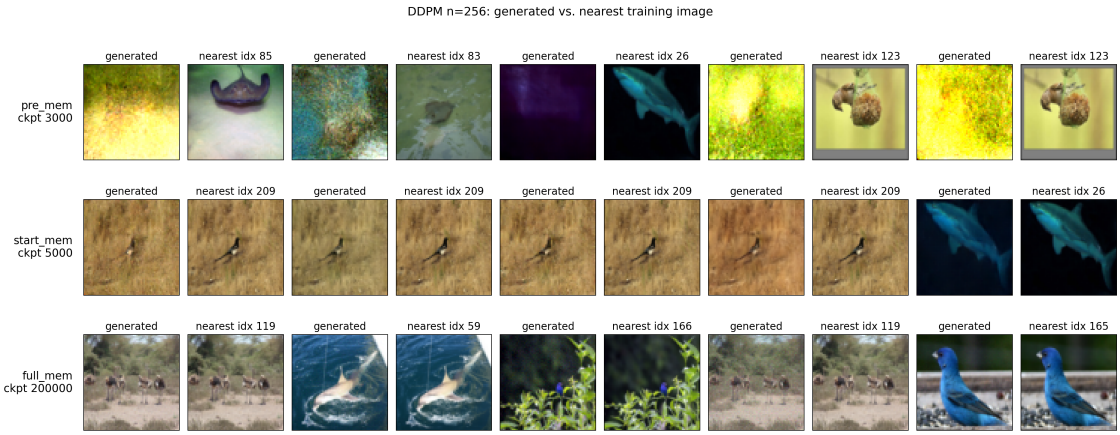


Figure 14: **Generated samples vs. nearest training image for ImageNet** ($n = 256$) **DDPM.** pre_mem: step 3,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 5,000, $f_{\text{mem}} = 0.04\%$ (4 copies of 10^4); the four copies, followed in the rightmost pair by the non-copy with the smallest gap ratio. full_mem: step 200,000, $f_{\text{mem}} = 95.5\%$, five randomly selected copies.

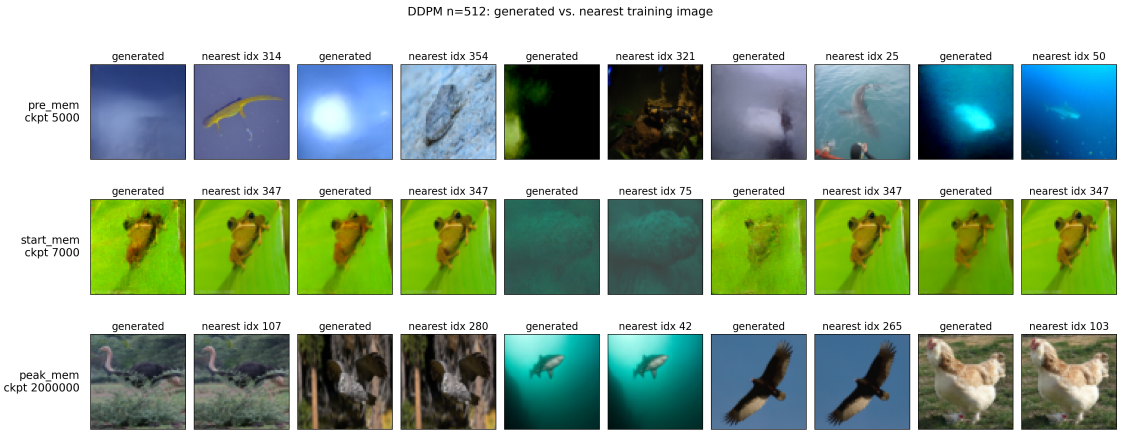


Figure 15: **Generated samples vs. nearest training image for ImageNet** ($n = 512$) **DDPM.** pre_mem: step 5,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 7,000, $f_{\text{mem}} = 0.03\%$ (3 copies of 10^4); the three copies, followed in the two rightmost pairs by the non-copies with the smallest gap ratios. peak_mem: step 2,000,000, $f_{\text{mem}} = 85.0\%$, five randomly selected copies. This model never reaches the full-memorization threshold.

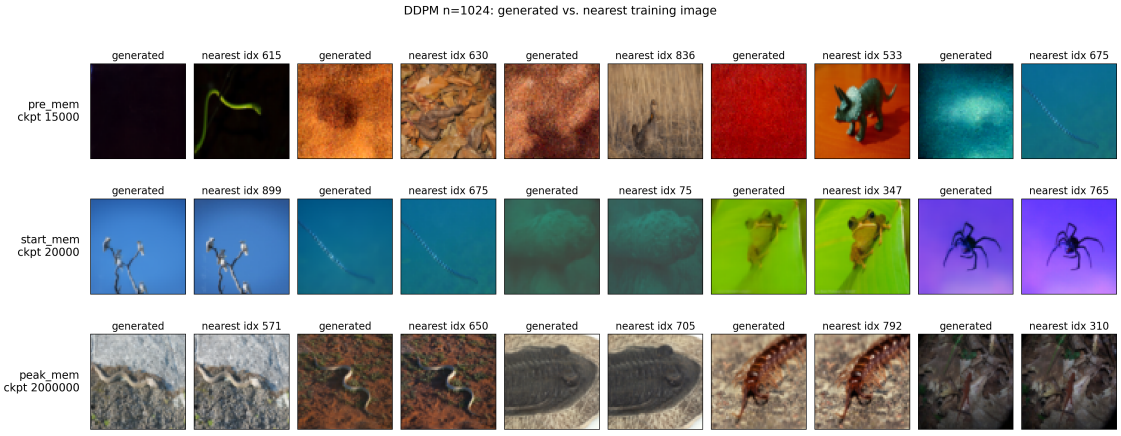


Figure 16: **Generated samples vs. nearest training image for ImageNet** ($n = 1024$) **DDPM.** pre_mem: step 15,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 20,000, $f_{\text{mem}} = 0.10\%$ (10 copies of 10^4), five randomly selected copies. peak_mem: step 2,000,000, $f_{\text{mem}} = 19.9\%$, five randomly selected copies. This model never reaches the full-memorization threshold.

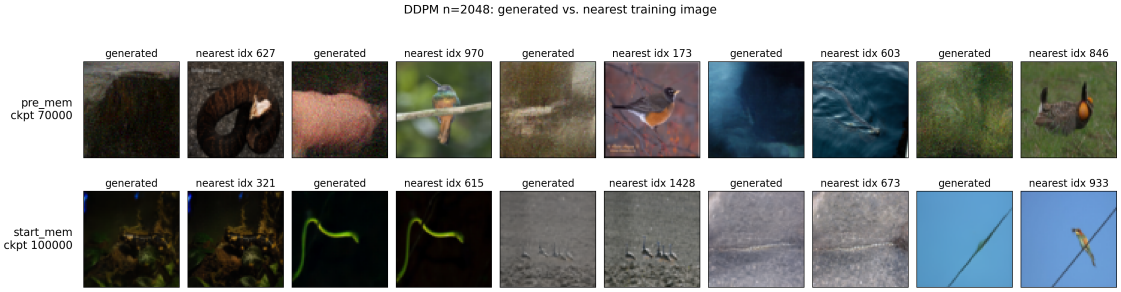


Figure 17: **Generated samples vs. nearest training image for ImageNet** ($n = 2048$) **DDPM.** pre_mem: step 70,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 100,000, $f_{\text{mem}} = 0.02\%$ (2 copies of 10^4); the two copies, followed in the three rightmost pairs by the non-copies with the smallest gap ratios. There is no full_mem row: f_{mem} peaks at 0.57%.

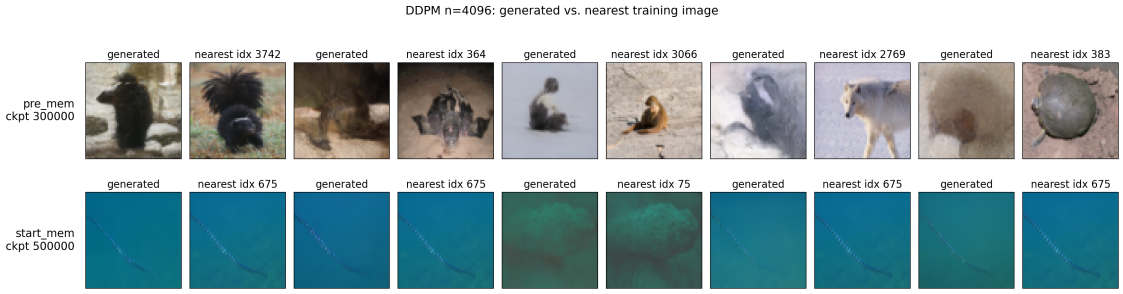


Figure 18: **Generated samples vs. nearest training image for ImageNet** ($n = 4096$) **DDPM.** pre_mem: step 300,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 500,000, an isolated checkpoint with $f_{\text{mem}} = 0.03\%$ (3 copies of 10^4); the three copies, followed in the two rightmost pairs by the non-copies with the smallest gap ratios. Here start_mem labels the isolated copy event. No sustained memorization is observed, and there is no full_mem row.

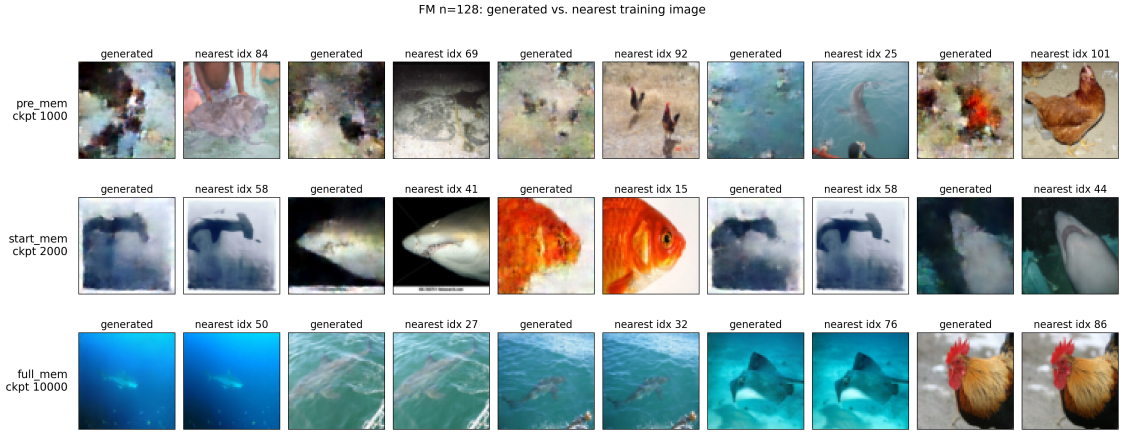


Figure 19: **Generated samples vs. nearest training image for ImageNet** ($n = 128$) **flow matching.** pre_mem: step 1,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 2,000, $f_{\text{mem}} = 0.10\%$ (10 copies of 10^4), five randomly selected copies. full_mem: step 10,000, $f_{\text{mem}} = 95.7\%$, five randomly selected copies.

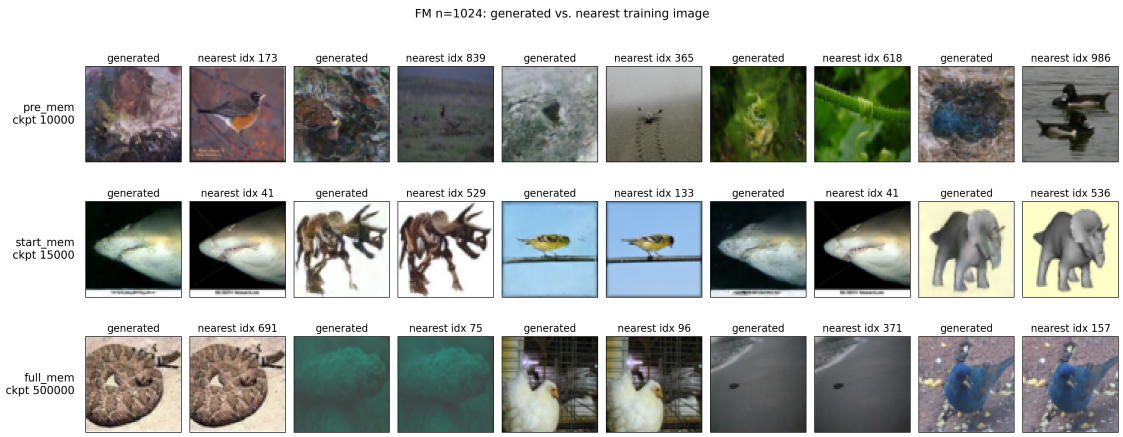


Figure 20: **Generated samples vs. nearest training image for ImageNet** ($n = 1024$) **flow matching.** pre_mem: step 10,000, $f_{\text{mem}} = 0$, five random samples. start_mem: step 15,000, $f_{\text{mem}} = 0.46\%$ (46 copies of 10^4), five randomly selected copies. full_mem: step 500,000, $f_{\text{mem}} = 96.2\%$, five randomly selected copies.

D Additional Experiments

We provide additional experiments on Imagenet and CelebA. Figures 21-25 plot the average training input loss curvature over diffusion time and fraction of generated samples that are considered memorized with memorization threshold $k = 1/3$ vs. training step for DDPM noise prediction and flow matching for the datasets CelebA with training set size $n \in \{128, 1024\}$ and Imagenet with training set size $n \in \{128, 1024, 10240\}$, respectively. DDPM and flow matching on CelebA with $n = 128$ is a memorizing model, since fraction memorization starts increasing rapidly to 100% around training step $\tau = 10^3$, whereas DDPM and flow matching on CelebA with $n = 10240$ is a generalizing model with fraction memorization increasing much later around training step $\tau = 10^5$ and never reaching full memorization of 100% before training ends. We observe that the average input loss curvature increases more drastically for the memorizing model as compared to the generalizing model. A similar trend can be seen for the DDPM and flow matching models for Imagenet as training size n increases.

For the flow matching model on CelebA with training size $n = 128$, Figure 26 shows the average training and validation input loss curvatures over diffusion time vs. training step, as well as the relative training and validation curvature gaps, which is calculated as $|\mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2] - \mathbb{E}_{x\sim\text{val.}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2]|/\mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2]$ and $|\mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2] - \mathbb{E}_{x\sim\text{val.}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2]|/\mathbb{E}_{x\sim\text{val}}[\|\nabla_x^2 \ell(\phi(\tau), x)\|_F^2]$, vs. training step. We observe that the training curvature separates from the validation curvature right before training step $\tau = 10^3$, which is around where memorization begins, as shown in Figure 21.

Figure 27 shows the fraction memorization score for DDPM noise prediction and flow matching on Imagenet ($n = 10240$) with different values for the memorization threshold $k \in \{1/3, 0.8, 0.9\}$. There is no trend in the nearest-neighbor gap ratios for DDPM even when k is increased. However, for flow matching, when k is large, the nearest-neighbor gap ratios increase as training continues. Nonetheless, this pattern in the gap ratios does not indicate that this model is a memorizing model, since for $k = 1/3$, the fraction memorized is approximately 0% for all training steps.

Figure 28 shows the average training curvature over diffusion time and fraction memorized with $k = 1/3$ vs. training step for DDPM sample prediction on Imagenet ($n = 10240$) as well as the fraction memorization with $k \in \{1/3, 0.8, 0.9\}$. This model is a generalizing model, since for $k = 1/3$, approximately 0% of the generated samples are considered memorized for all training steps, even though for large values of k, the nearest-neighbor gap ratios increase as training steps increase. We also observe that the average input loss curvature barely changes across training steps, validating the connection between memorization and curvature.

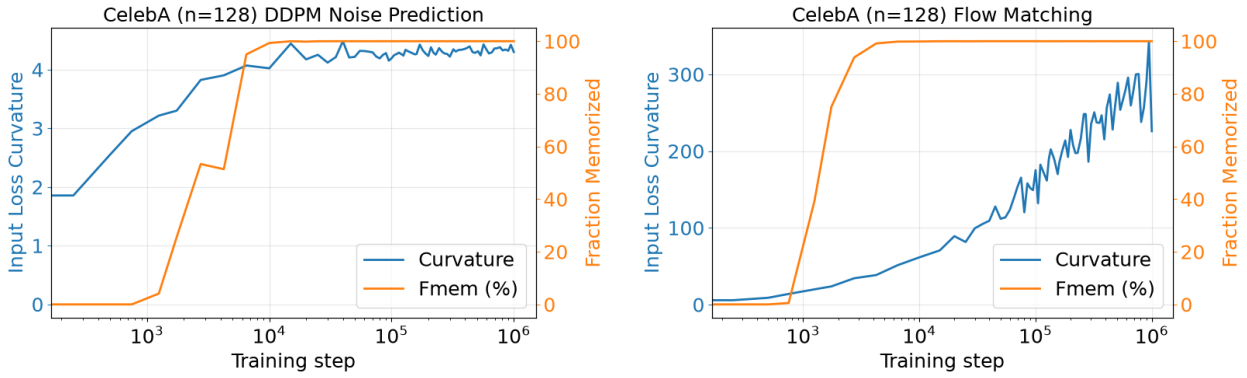
In Figure 29, the average training input loss curvature over diffusion time vs. training step is plotted for DDPM noise prediction and flow matching on the CelebA and Imagenet datasets with training set size $n = 10240$. CelebA has larger curvature values than Imagenet for both models, which correspond to higher memorization scores as seen in Figures 22 and 25. In particular, DDPM and flow matching for CelebA ($n = 10240$) have fraction memorization scores of around 23% and 40%, respectively, at the end of training, whereas DDPM and flow matching for Imagenet ($n = 10240$) have fraction memorization scores of around 0% and 0.1%, respectively, at the end of training.

Figure 30 compares two DDPM noise prediction models on Imagenet, one that is trained on a training set of size $n = 1024$ and another that is trained on the same training set with the top 10 memorized samples from the full $n = 1024$ model removed. The figure plots fraction memorization and average training curvature over diffusion time vs. training step for both the full and leave-10-out models. The leave-10-out model has a lower memorization score at the end of training and lower input loss curvature for all training steps, indicating that it is worthwhile to remove the top memorized samples from a model to improve generalization. Figure 31 shows similar results for flow matching. We repeat these experiments for Imagenet $n = 128$ in Figures 32 and 33. Figures 34 and 35 plot the difference between the per sample losses of the full flow matching models and the leave-10-out models vs. training step, as well as the per sample curvatures of the leave-10-out models, for each of the top 10 left-out samples.

E Additional ImageNet DDPM experiments

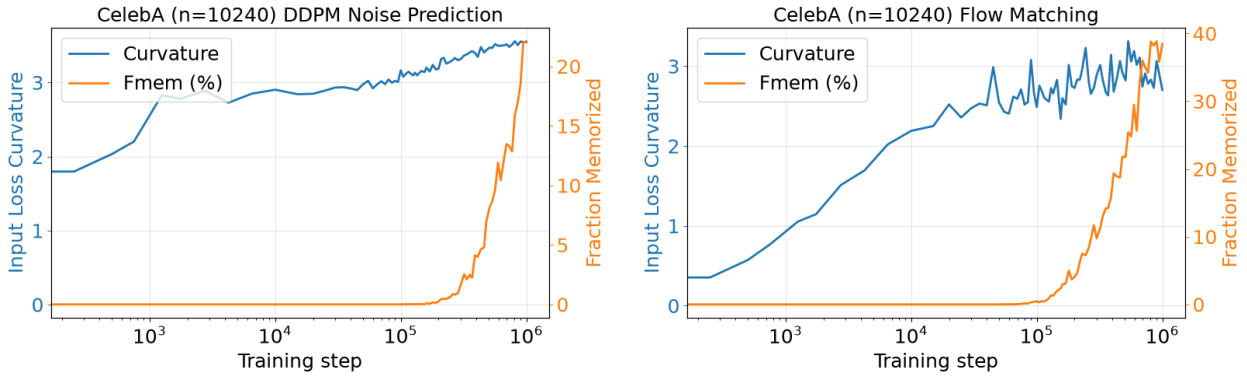
E.1 Experimental Details

We use 64×64 RGB ImageNet images and nested training sets of size $n \in \{128, 256, 512, 1024, 2048, 4096\}$, taking the first n images in a fixed dataset ordering. We evaluate all models on the same disjoint set of $1{,}024$ held-out images. Each training size corresponds to one DDPM trained with the noise-prediction objective. The U-Net has base width 64 and channel multipliers $(1, 2, 4, 4)$. We use Adam with learning rate 10^{-4} and a linear noise schedule with $1{,}000$ diffusion steps. We train all models for 10^6 updates and continue training those with $n \in \{512, 1024, 2048, 4096\}$ for another 10^6 updates to examine their behavior over longer training.



(a) (b)

Figure 21: **Curvature and memorization for CelebA (n=128)**: average input loss curvature over diffusion time and training set and fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step for (a) DDPM noise prediction and (b) flow matching.



(a) (b)

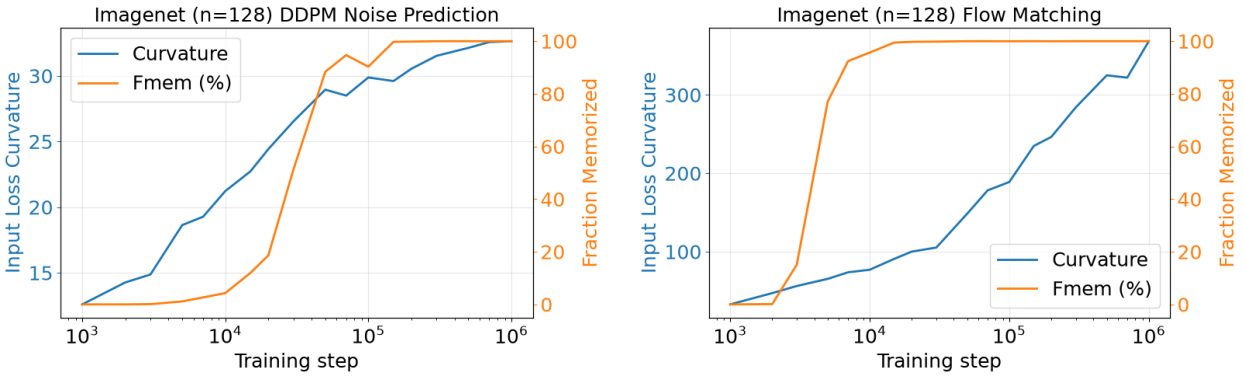
Figure 22: **Curvature and memorization for CelebA (n=10240)**: average input loss curvature over diffusion time and training set and fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step for (a) DDPM noise prediction and (b) flow matching.

For the curvature, gradient, and loss measurements, we convert the predicted noise to a clean-image estimate $\hat{x}_0$ and evaluate the mean squared reconstruction error

$$\ell_\tau(x; t, \epsilon) = \frac{1}{d} \left\| \hat{x}_0(x_t, t; \phi(\tau)) - x \right\|_2^2, \qquad d = 3 \cdot 64^2,$$

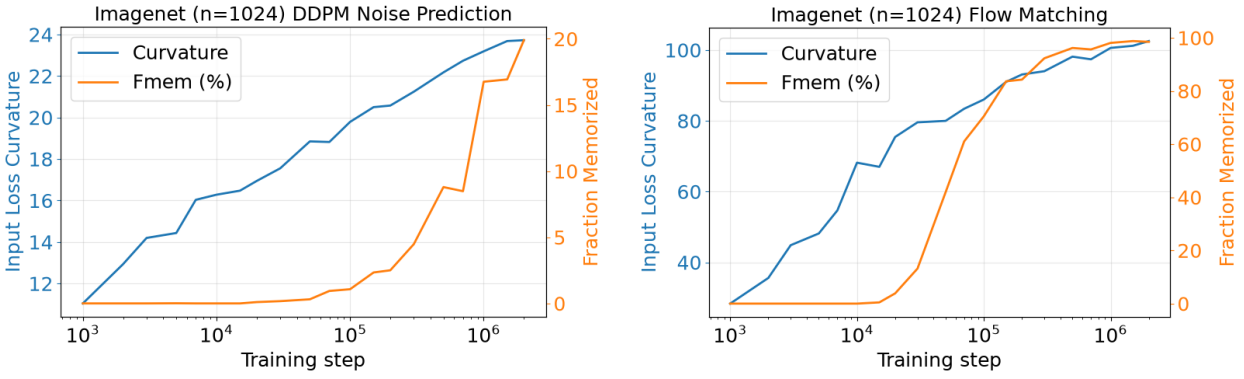
where d is the number of scalar image values across the three RGB channels, so $1/d$ averages the squared error over all channels and pixels. Here x is the clean image after the input normalization used for training, and x_t is its noisy version at diffusion step t, obtained using Gaussian noise ϵ. The training checkpoint is indexed by τ, and $\phi(\tau)$ denotes the model parameters at that checkpoint. Derivatives are taken with respect to x, including its contribution through x_t. We estimate the input-Hessian trace using 32 Monte Carlo draws per image. Each draw samples t uniformly from $\{1, \ldots, 999\}$, Gaussian forward noise ϵ, and a Rademacher vector v, and evaluates $v^\top \nabla_x^2 \ell_\tau(x; t, \epsilon) v$. The gradient curves average $\|\nabla_x \ell_\tau(x; t, \epsilon)\|_2$ over 32 draws, and the loss curves average ℓ_τ over 64 draws. Training means use all n images; validation means use the fixed held-out set. These measurements do not require generating samples from the model.

At each checkpoint used to evaluate memorization, we generate 10,000 images using 100 deterministic DDIM steps ($\eta = 0$). A generated image is classified as a copy when the ratio of its nearest to second-nearest training-image distances is below $1/3$. We use Euclidean distances in the normalized image space used for training. The copy count $F(i, \tau)$ is the number of generated copies assigned to training image i at checkpoint τ, and $f_{\text{mem}}(\tau)$ is the fraction of generations classified as copies.



(a) (b)

Figure 23: **Curvature and memorization for Imagenet (n=128)**: average input loss curvature over diffusion time and training set and fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step for (a) DDPM noise prediction and (b) flow matching.

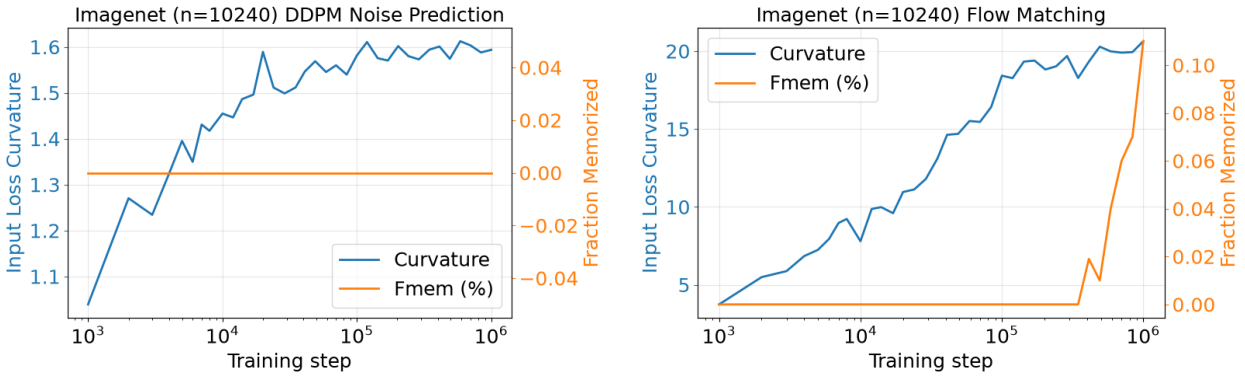


(a) (b)

Figure 24: **Curvature and memorization for Imagenet (n=1024)**: average input loss curvature over diffusion time and training set and fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step for (a) DDPM noise prediction and (b) flow matching.

E.2 Supporting Figures

Figure 37 compares the mean curvature of training images copied at a checkpoint with that of the remaining training images. The copied group generally has higher curvature, although the separation varies with training size and checkpoint. Figure 38 compares the maximum copy count with the relative train–validation curvature gap. Smaller training sets tend to exhibit both larger maximum copy counts and larger curvature gaps.



(a) (b)

Figure 25: **Curvature and memorization for Imagenet (n=10240)**: average input loss curvature over diffusion time and training set and fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step for (a) DDPM noise prediction and (b) flow matching.

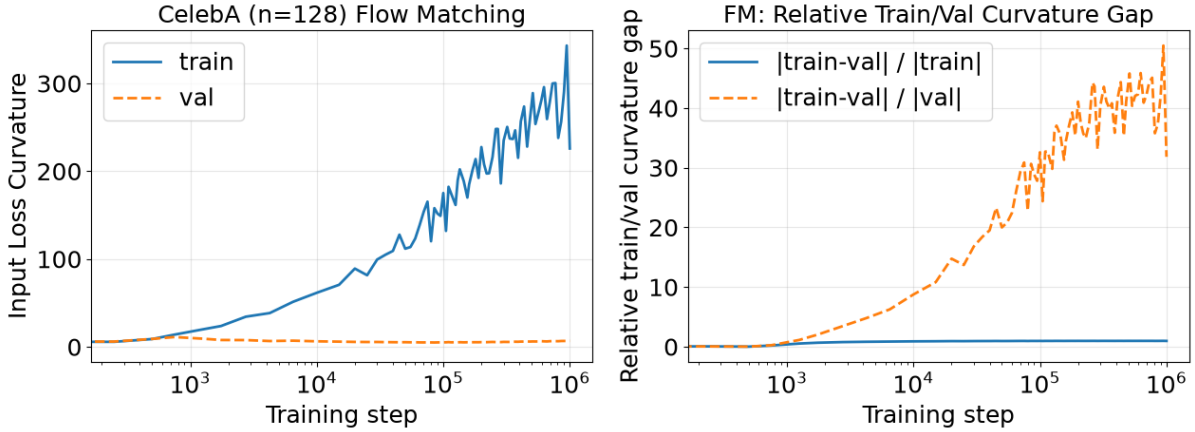
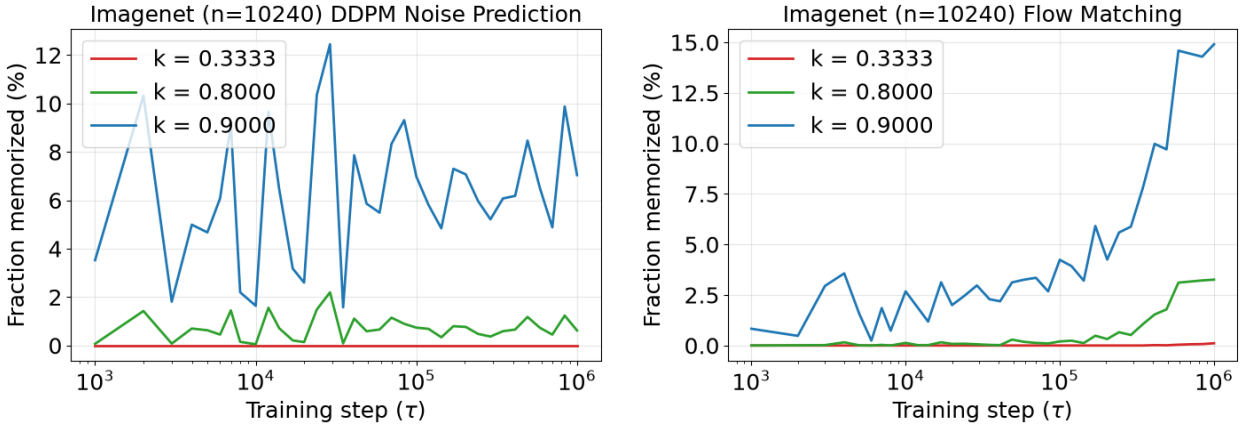
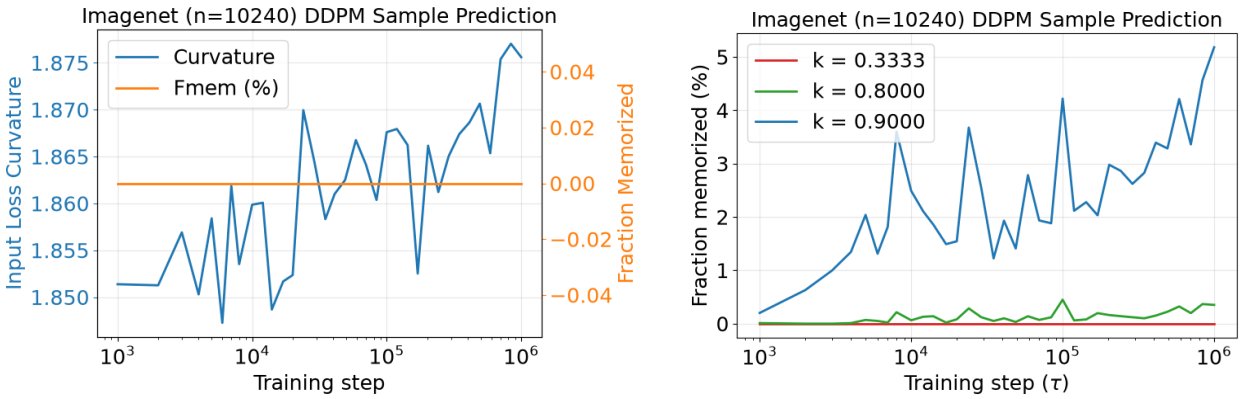


Figure 26: **Training vs. validation curvature for CelebA (n=128) flow matching**: (left) average input loss curvature over diffusion time and training (blue) / validation (orange) set, (right) relative training/validation curvature gap calculated as $|\mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2\ell(\phi(\tau),x)\|_F^2] - \mathbb{E}_{x\sim\text{val.}}[\|\nabla_x^2\ell(\phi(\tau),x)\|_F^2]|/\mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2\ell(\phi(\tau),x)\|_F^2]$ (blue) and $|\mathbb{E}_{x\sim\text{train}}[\|\nabla_x^2\ell(\phi(\tau),x)\|_F^2] - \mathbb{E}_{x\sim\text{val.}}[\|\nabla_x^2\ell(\phi(\tau),x)\|_F^2]|/\mathbb{E}_{x\sim\text{val}}[\|\nabla_x^2\ell(\phi(\tau),x)\|_F^2]$ (orange).



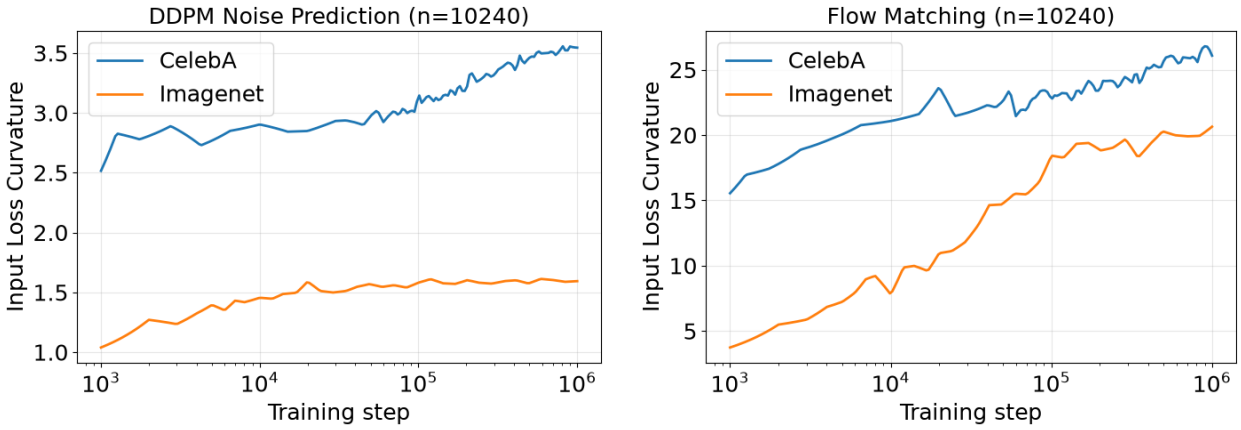
(a) (b)

Figure 27: **Memorization for Imagenet (n=10240)**: fraction memorized (%) with memorization threshold $k \in \{1/3, 0.8, 0.9\}$ vs. training step for (a) DDPM noise prediction and (b) flow matching.



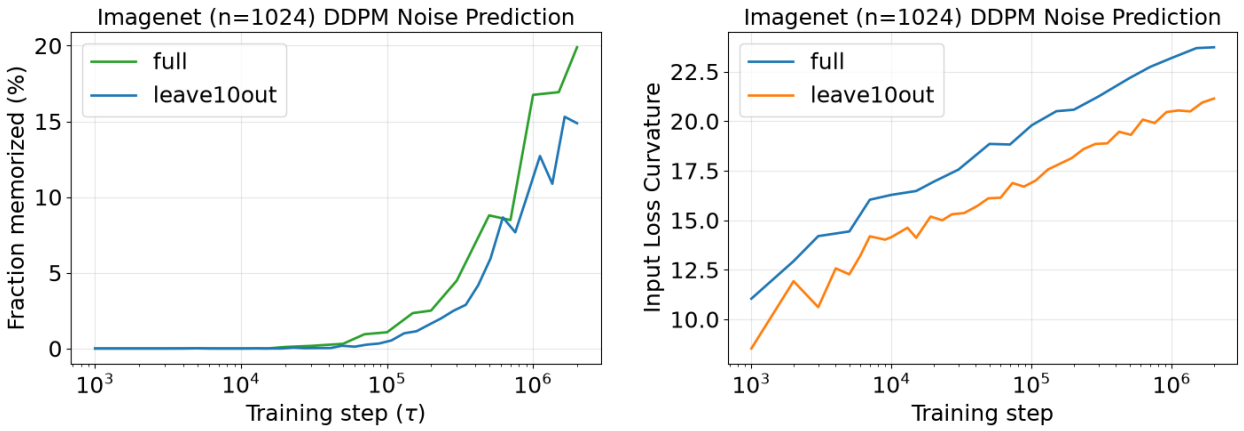
(a) (b)

Figure 28: **Curvature and memorization for Imagenet (n=10240) DDPM Sample Prediction**: (a) average input loss curvature over diffusion time and training set and fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step, and (b) fraction memorized (%) with memorization threshold $k \in \{1/3, 0.8, 0.9\}$ vs. training step.



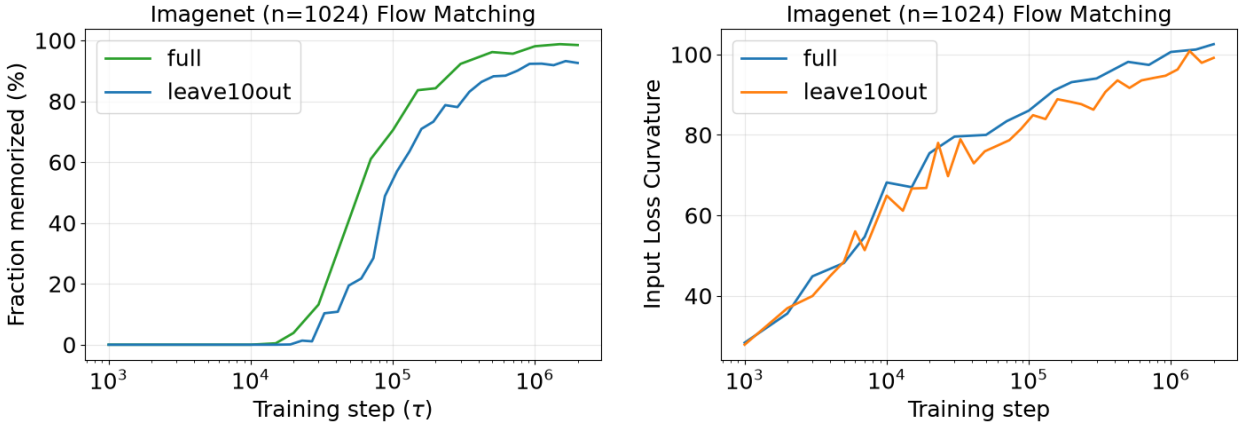
(a) (b)

Figure 29: **CelebA vs. Imagenet (n=10240) Curvature**: average input loss curvature over diffusion time and training set vs. training step for (a) DDPM noise prediction and (b) flow matching.



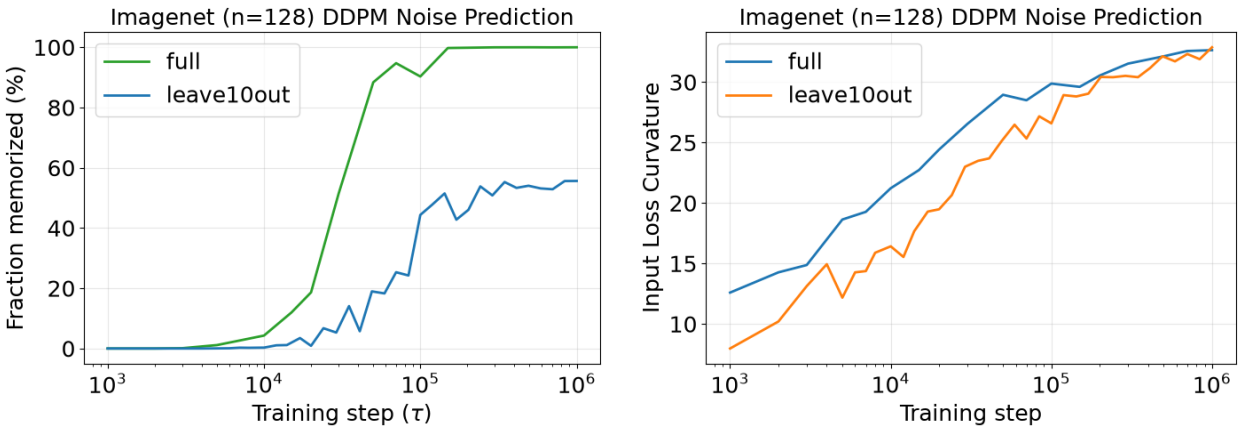
(a) (b)

Figure 30: **Memorization and curvature for Imagenet (n=1024) DDPM noise prediction - full vs. leave-10-out training**: (a) fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step and (b) average input loss curvature over diffusion time and training set vs. training step for full $n = 1024$ training set and leave-10-out training set with the top 10 memorized training samples removed.



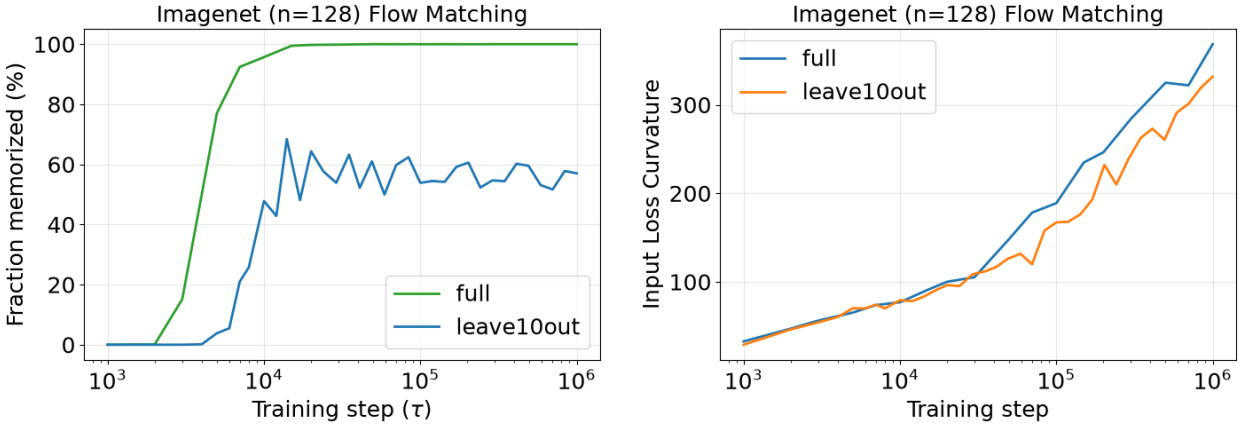
(a) (b)

Figure 31: **Memorization and curvature for Imagenet (n=1024) flow matching - full vs. leave-10-out training**: (a) fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step and (b) average input loss curvature over diffusion time and training set vs. training step for full $n = 1024$ training set and leave-10-out training set with the top 10 memorized training samples removed.



(a) (b)

Figure 32: **Memorization and curvature for Imagenet (n=128) DDPM noise prediction - full vs. leave-10-out training**: (a) fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step and (b) average input loss curvature over diffusion time and training set vs. training step for full $n = 128$ training set and leave-10-out training set with the top 10 memorized training samples removed.



(a) (b)

Figure 33: **Memorization and curvature for Imagenet (n=128) flow matching - full vs. leave-10-out training**: (a) fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step and (b) average input loss curvature over diffusion time and training set vs. training step for full $n = 128$ training set and leave-10-out training set with the top 10 memorized training samples removed.

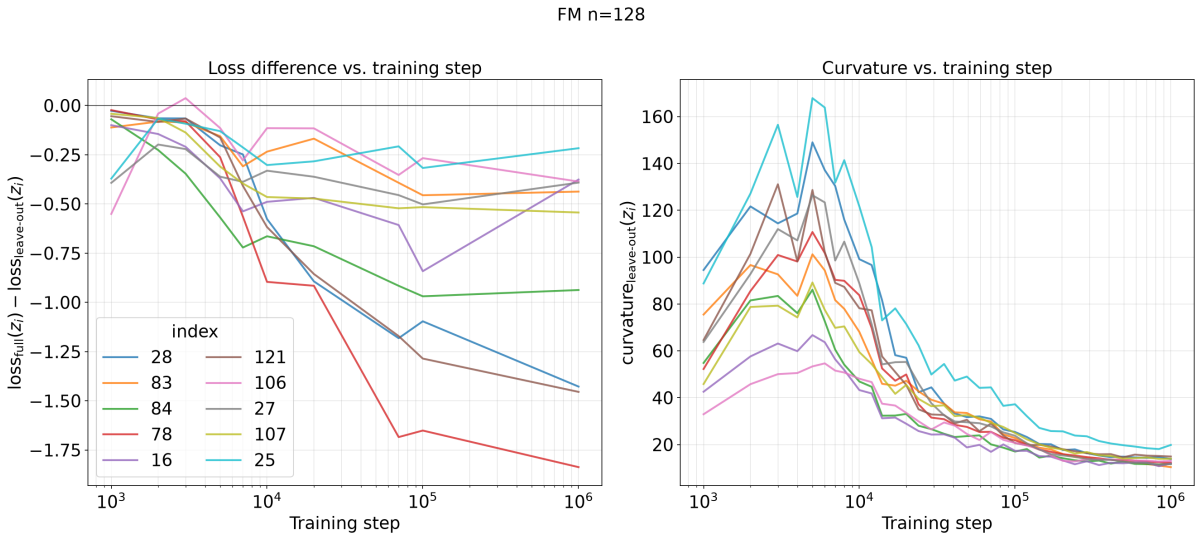


Figure 34: **Loss difference and curvature for Imagenet (n=128) flow matching - leave-10-out-training**: (left) difference between per sample loss of full model and leave-10-out model vs. training step for each of the 10 left-out training samples, (right) per sample curvature of the leave-10-out model vs. training step for each of the 10 left-out training samples.

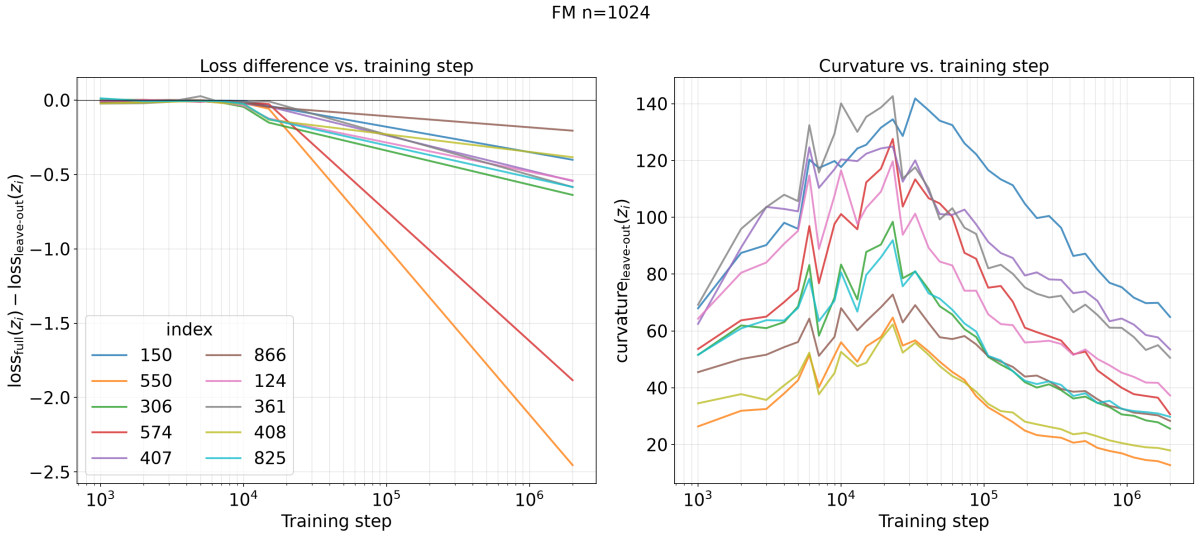
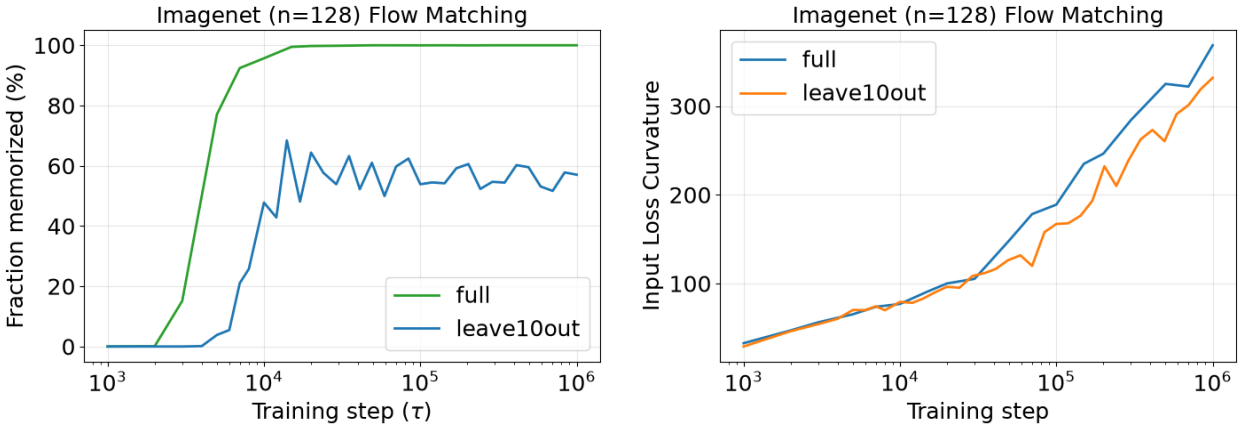


Figure 35: **Loss difference and curvature for Imagenet (n=1024) flow matching - leave-10-out-training**: (left) difference between per sample loss of full model and leave-10-out model vs. training step for each of the 10 left-out training samples, (right) per sample curvature of the leave-10-out model vs. training step for each of the 10 left-out training samples.



(a) (b)

Figure 36: **Memorization and curvature for Imagenet (n=128) flow matching - full vs. leave-10-out training**: (a) fraction memorized (%) with memorization threshold $k = 1/3$ vs. training step and (b) average input loss curvature over diffusion time and training set vs. training step for full $n = 128$ training set and leave-10-out training set with the top 10 memorized training samples removed.

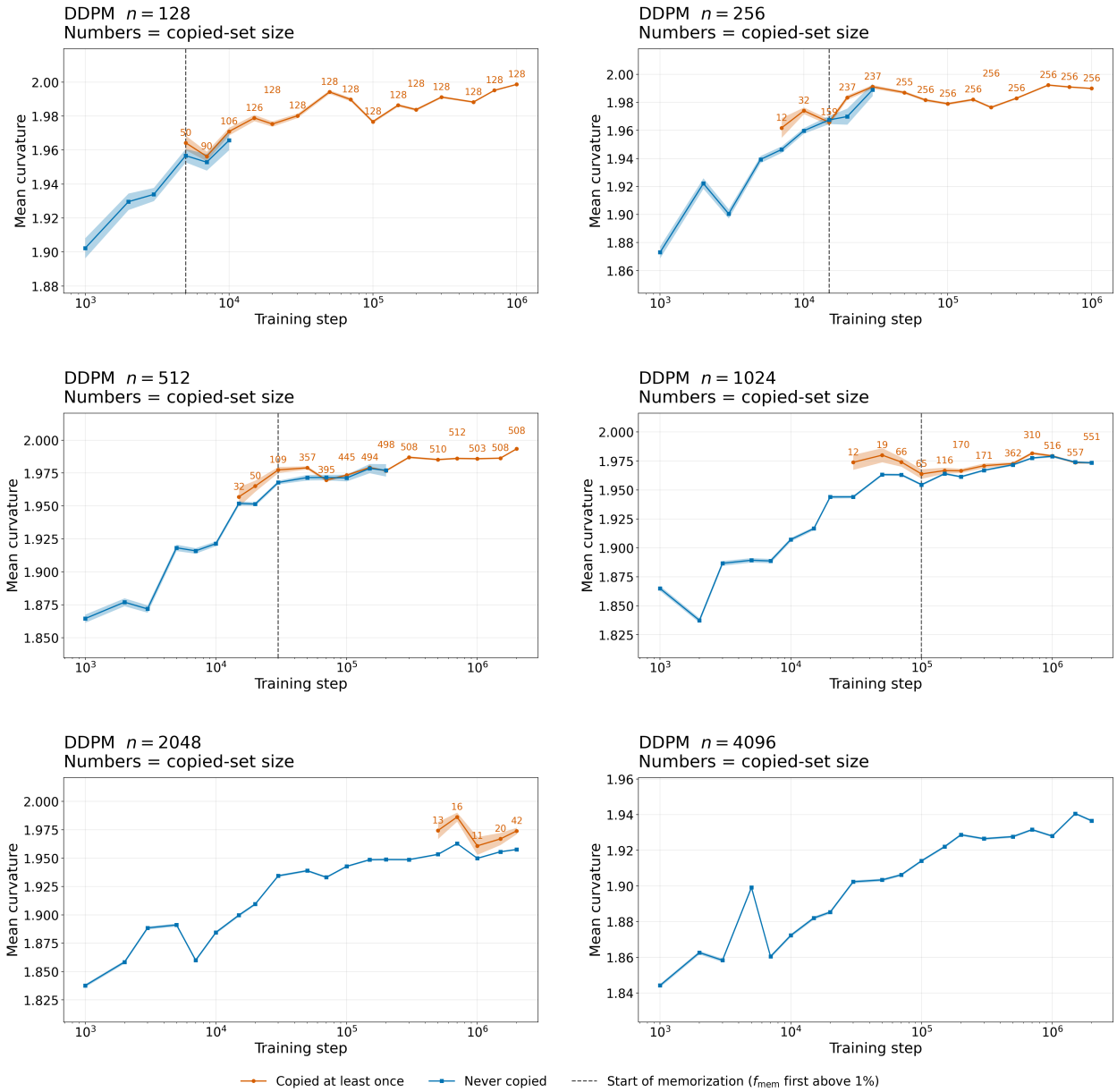


Figure 37: **Curvature of copied and non-copied training images.** Each panel shows one ImageNet DDPM training size. Orange and blue curves average the input-Hessian trace over images with $F(i,\tau) > 0$ and $F(i,\tau) = 0$, respectively. Group membership is determined separately at each checkpoint; "never copied" in the legend refers only to that checkpoint's generations. Shading shows one standard error across images in the group, and orange numbers give the copied-group size. A group is plotted only when it contains at least ten images. Dashed vertical lines mark the first evaluated checkpoint with $f_{\text{mem}} > 1\%$, used as a reference threshold.

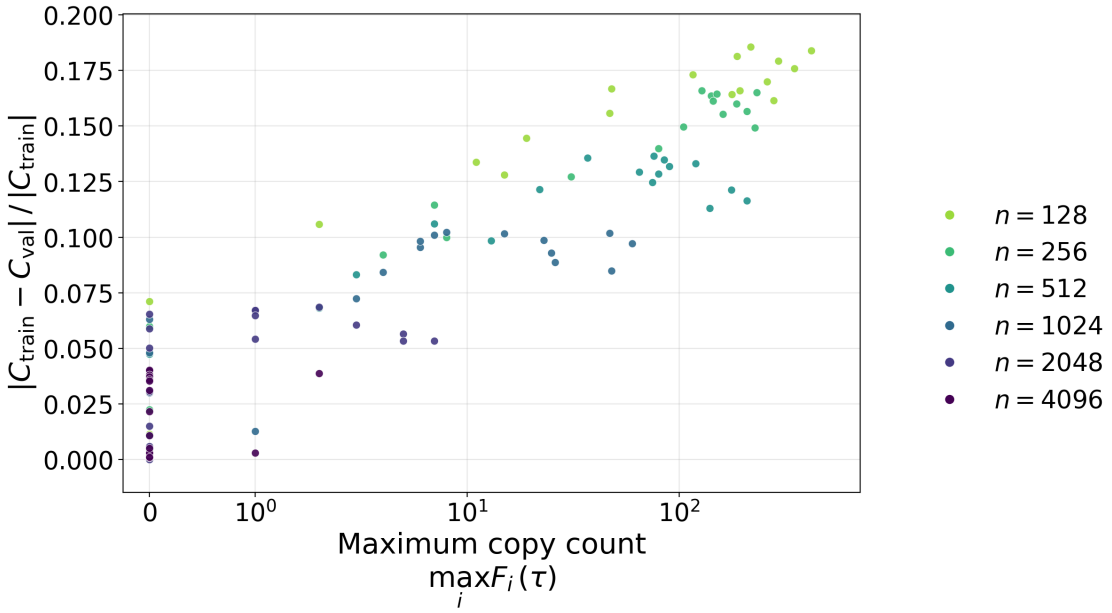


Figure 38: **Maximum copy count and relative curvature gap.** Each point represents one evaluated checkpoint of an ImageNet DDPM; colors indicate training-set size. The horizontal coordinate is $\max_i F(i, \tau)$ among $10{,}000$ generations. The vertical coordinate is $|C_{\text{train}}(\tau) - C_{\text{val}}(\tau)|/|C_{\text{train}}(\tau)|$, where the two means use all training images and the same $1{,}024$ held-out images, respectively. Curvature is the input-Hessian trace estimated as described above. The horizontal scale is logarithmic above one and linear near zero to retain checkpoints with no observed copies.