On the Discrepancy and Connection between Memorization and Generation in Diffusion Models

Hanyu Wang [1] **Yujin Han** [2] **Difan Zou** [3]

Abstract

Diffusion models (DMs), as a state-of-the-art generative modeling method, have enjoyed tremendous success in multiple generating tasks. However, the memorization behavior of DMs, that the generation replicates the training data, raises serious privacy concerns and contradicts the actual generalizability of DMs. These prompt us to delve deeper into the generalizability and memorization of DMs, particularly in cases where the closed-form solution of DMs' score function can be explicitly solved. Through a series of comprehensive experiments, we demonstrate the discrepancies and connections between the optimal score and the trained score, noting that the trained one is smoother, which benefits the generalizability of DMs. We also further explore how mixing the optimal score with the trained score during the sampling phase affects generation. Our experimental findings provide novel insights into the understanding of DMs' generalizability.

1. Introduction

Diffusion models (DMs) have demonstrated exceptional capability in generating high-quality images (Ho et al., 2020; Song et al., 2020; Vahdat et al., 2021; Dhariwal & Nichol, 2021), audio (Liu et al., 2023; Yang et al., 2024), and videos (Ho et al., 2022). However, recent works have reported the memorization behavior of DMs (Carlini et al., 2023; Gu et al., 2023; Wen et al., 2023; Yoon et al., 2023; Zhang et al., 2023), specifically, generating samples that replicate training data. The memorization phenomenon contradicts the generalization ability of DMs (Kadkhodaie et al., 2023; Oko et al., 2023; Li et al., 2024) and also raises concerns about privacy issues when applied (Carlini et al., 2023). Therefore, this memorization phenomenon needs deep investigation.

There has been a line of work (Somepalli et al., 2023a; Carlini et al., 2023; Wen et al., 2023; Somepalli et al., 2023b; Yi et al., 2023; Gu et al., 2023) attempting to understand the memorization behavior of DMs. However, they primarily examine the effects of factors like training dataset size (Somepalli et al., 2023a), prompts (Wen et al., 2023; Somepalli et al., 2023b), and initializations (Wen et al., 2023) on DMs' memorization, rather than focusing on more fundamental aspects, such as the influence of DMs' training processes on generalizability. Specifically, for the typical denoising score matching loss of DMs, the closed-form optimal solution of the denoising score suggests the necessity of DMs replicating training data (Yi et al., 2023; Gu et al., 2023), which contradicts the generalizability observed in the denoising scores obtained from actual DMs training. Therefore, it naturally leads us to ask:

What are the discrepancies and connections between these two scores regarding memorization and generation?

To address the question, our study primarily compares the oracle true score and the trained one from various perspectives. The comparison demonstrates that the data generated by both can minimize the denoising loss, while the trained score is smoother, thereby facilitating generalizability. We further investigate the effects of mixing the optimal and trained denoising scores during sampling on the memorization behavior of DMs. We find that introducing the optimal score exacerbates the memorization of DMs, with even late-stage replacement leading to the exact replication of training data. Our contributions are summarized as follows:

- We conduct various experiments to compare the optimal and trained denoising scores, emphasizing their discrepancies and connections on generalizability. This offers an intuitive explanation for the superior generalizability of DMs based on trained scores compared to those based on oracle scores.
- We mix the optimal denoising score with the trained denoising score during the sampling phase and observe that the introduction of the optimal denoising score exacerbates the memorization behavior of DMs.

[1]Department of Mathematics, University of Science and Technology of China [2]Department of Computer Science, The University of Hong Kong [3]Department of Computer Science & Institute of Data Science, The University of Hong Kong. Correspondence to: Difan Zou <dzou@cs.hku.hk>.

Published at ICML 2024 Workshop on Foundation Models in the Wild.

2. Preliminary

In this section, we will introduce the notations and problem settings that are commonly used in the following section.

Notations. We refer to $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ as the trained score and $\boldsymbol{\epsilon}^*$ as the nonparametric optimal score. We denote the Jacobian matrix of a vector-valued function $\mathbf{f}$ with respect to $\mathbf{x}$ as $\mathcal{J}\mathbf{f}(\mathbf{x})$. $\mathcal{S}(\mathbf{x})$ denotes the softmax vector of $\mathbf{x} = (x_1, \ldots, x_n)$. $\langle \mathbf{x}, \mathbf{y} \rangle$ refers to inner product of n-dimensional vector $\mathbf{x}, \mathbf{y}$ which is defined as $\sum_{i=1}^n x_i y_i$. $\{\mathbf{x}_0^i\}_{i=1}^n$ refers to the training set of diffusion models. All the norms $\|\cdot\|$ are L_2 norm.

Diffusion Models. Inspired by thermodynamics, DMs are a two-stage framework that includes a forward process (mapping data distribution p_0 to noise distribution q_0) and a reverse process (transforming noise distribution q_0 back to data p_0). We consider DDPM (Ho et al., 2020) in our study, the forward process is a Markov chain whose transition kernel is Gaussian perturbation, which gradually adds standard Gaussian noise to the data sample $\mathbf{x}_0$. At time step t, the transition kernel is

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{1}$$

where $\beta_t \in (0, 1)$ is a hyperparameter chosen prior training.

Conversely, the reverse process aims to remove the added noise and recover the original data using a learnable transition kernel, described as follows:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2}$$

where θ denotes trainable model parameters and the statistics-mean $\boldsymbol{\mu}_\theta$ and variance $\boldsymbol{\Sigma}_\theta$-are parameterized by deep neural networks. With a trained learnable transition kernel, data can be generated by sampling from the standard Gaussian by following $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ till $t = 1$.

Trained Denosing Score. The remaining task is training the reverse transition kernel. Derived from evidence lower bound (ELBO), this goal can be achieved by minimizing the following weighted sum of denoising score network $\boldsymbol{\epsilon}_\theta$ objectives (Ho et al., 2020):

$$\mathbb{E}_{t\sim\mathcal{U}(1,T),\mathbf{x}_0\sim q_0,\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\lambda(t)\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2] \tag{3}$$

where $\lambda(t)$ is a positive weighting function for better sample quality (Ho et al., 2020; Yang et al., 2023) and with the time length T, $\mathcal{U}(1, T)$ is a uniform distribution over the set $\{1, \cdots, T\}$. $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is the trained denosing network to predict the ground truth noise $\boldsymbol{\epsilon}_t$ given $\mathbf{x}_t$ and t. More details about DDPM are included in Appendix D.

Optimal Denosing Score. Given the ground truth noise $\boldsymbol{\epsilon}$ satisfies the standard Gausssian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, the closed-form optima of empirical counterpart of Equation (3) can be derived as follows (Yi et al., 2023; Gu et al., 2023):

$$\boldsymbol{\epsilon}^*(\mathbf{x}_t, t) = \frac{\mathbf{x}_t}{\sqrt{1-\bar{\alpha}_t}} - \left(\frac{\sqrt{\bar{\alpha}_t}}{\sqrt{1-\bar{\alpha}_t}}\right) \sum_{i=1}^n \frac{\exp\left(-\frac{\|\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\|^2}{2(1-\bar{\alpha}_t)}\right)\mathbf{x}_0^i}{\sum_{i=1}^n \exp\left(-\frac{\|\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\|^2}{2(1-\bar{\alpha}_t)}\right)}. \tag{4}$$

where $\bar{\alpha}_t := \prod_{1\le s\le t} \alpha_s$ and $\alpha_t := 1 - \beta_t$. The oracle solution $\boldsymbol{\epsilon}^*$ can actually be regarded as a linear combination of the training data $\mathbf{x}_0$ given $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, suggesting the memorization of DDPM towards $\mathbf{x}_0$.

In our study, we aim to understand the memorization of DDPM by comparing the trained solution $\boldsymbol{\epsilon}_\theta$ in (3) with the oracle solution $\boldsymbol{\epsilon}^*$ in (4) from various perspectives.

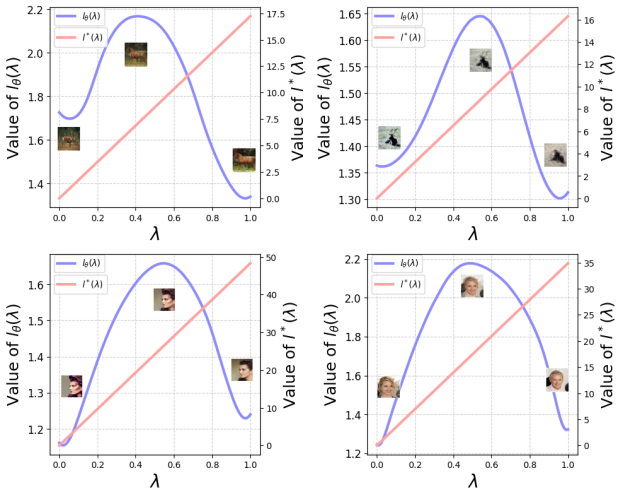


Figure 1: Visualization of $l_\theta^t(\lambda)$ and $l_*^t(\lambda)$ on CIFAR10 and CelebA with fixed $t = 10$. $\lambda = 1$ refers to generated image while $\lambda = 0$ refers to its nearest neighbor in the training set. We show pictures of $\mathbf{z}_0(0), \mathbf{z}_0(0.5), \mathbf{z}_0(1)$ from left to right.

3. Main Results

In this section, we show the main results of our work. First we explore the similarity and difference between the trained score and nonparametric optimal score from the following three aspects. 1) We propose a criterion inspired by the expression of nonparametric optimal score and verify that generated data are local minima. 2) We utilize pseudo difference quotient to compare the local geometric features of the two score functions. 3) We adopt SVD on Jacobian matrix to explore the information carried by the two score functions. Afterwards, we mix the two scores into one sampling process to check the memorization status.

3.1. Generated Data Are Local Minimum

First we focus on the closed form expression of $\boldsymbol{\epsilon}^*$ in (4). The interested softmax part inspires us to extract the same part from the trained score by a similar definition, and we

denote it $\hat{x}$:

$$\hat{x}_\theta(\mathbf{x}, t) \triangleq \frac{\mathbf{x} - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}, t)}{\sqrt{\bar{\alpha}_t}}, \tag{5}$$

$$\hat{x}^*(\mathbf{x}, t) \triangleq \frac{\mathbf{x} - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}^*(\mathbf{x}, t)}{\sqrt{\bar{\alpha}_t}}. \tag{6}$$

In particular, the former is popular as the predicted original data(Ho et al., 2020) while the latter is exactly

$$\sum_{i=1}^{n} \frac{\exp\left(-\frac{\left\|\mathbf{x}-\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\right\|^2}{2(1-\bar{\alpha}_t)}\right)\mathbf{x}_0^i}{\sum_{i=1}^{n}\exp\left(-\frac{\left\|\mathbf{x}-\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\right\|^2}{2(1-\bar{\alpha}_t)}\right)}. \tag{7}$$

When t is small, (7) is a certain training sample $\mathbf{x}_0^i$ which is determined by

$$i = \arg\min_j \left\|\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0^j\right\|^2. \tag{8}$$

Thus with the goal of investigating the relationship between a generated new image $\mathbf{x}_0$ and its pixel-wise nearest neighbor $\mathbf{y}_0$ in the training set, we consider the interpolation of their noisy version: $\mathbf{z}_t(\lambda) = \lambda\mathbf{y}_t + (1-\lambda)\mathbf{x}_t, \lambda \in [0, 1]$. To assess the capability of the trained score to restore the interpolation to original clean data, we introduce a new criterion as follows

$$l_\theta^t(\lambda) \triangleq \left\|\hat{x}_\theta(\mathbf{z}_t(\lambda), t) - \mathbf{z}_0(\lambda)\right\|, \tag{9}$$

$$l_*^t(\lambda) \triangleq \left\|\hat{x}^*(\mathbf{z}_t(\lambda), t) - \mathbf{z}_0(\lambda)\right\|. \tag{10}$$

Since the latter is approximately $\left\|\mathbf{x}_0^i - \mathbf{z}_0(\lambda)\right\|$, we pay more attention to $l_\theta^t(\lambda)$. When t is small, it is natural for $l_\theta^t(0)$ to be small because it represents the ability of the trained score network to denoise data from the training set that is perturbed by small-scale noise. But we surprisingly find that $l_\theta^t(1)$ is as small as $l_\theta^t(0)$ while $l_\theta^t(\lambda), \lambda \in (0, 1)$ is bigger, demonstrating that the trained score is able to denoise both training samples and generated data but is impotent faced with interpolation. Besides, l_*^t shows a monotonically increasing trend, suggesting that $\boldsymbol{\epsilon}^*$ does not restore the noisy generated data to clean ones.

Figure 1 shows the graph of $l_\theta^t(\lambda)$ with respect to λ and detailed experimental results are in Appendix C. We can summarize that generated images, similar to training samples, reside in some local minima regarding l_θ^t.

3.2. First Order Behavior

In this section, we analyze the first order behavior of $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and $\boldsymbol{\epsilon}^*$ given fixed t. We use difference quotient to analyze local properties on the line connecting two training data and Jacobian matrix to compare the information carried by the two score functions at certain training samples. We find that trained score is much smoother than nonparametric oracle score around the training samples, leading to the potential of generating new samples.

Difference Quotient. Difference quotient is a computationally friendly mathematical tool to gain insights into the local behavior of certain functions along interested directions. We leave more detailed discussions about this tool in Appendix B.1 and here we directly use it on score functions at different time step t, especially small ones because they contain more information about the original data distribution and thus are harder to learn for a neural network. With the goal of figuring out the local behavior along the line connecting any two training samples, we again consider the interpolation. Specifically, we select two samples $\mathbf{x}_0, \mathbf{y}_0$ from the training set and obtain the noisy version $\mathbf{x}_t, \mathbf{y}_t$ from the forward process. For the optimal score and trained score, we obtain the corresponding difference quotient:

$$\frac{\left\|\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\lambda\mathbf{x}_t + (1-\lambda)\mathbf{y}_t, t)\right\|}{(1-\lambda)\left\|\mathbf{x}_t - \mathbf{y}_t\right\|}, \tag{11}$$

$$\frac{\left\|\boldsymbol{\epsilon}^*(\mathbf{x}_t, t) - \boldsymbol{\epsilon}^*(\lambda\mathbf{x}_t + (1-\lambda)\mathbf{y}_t, t)\right\|}{(1-\lambda)\left\|\mathbf{x}_t - \mathbf{y}_t\right\|}. \tag{12}$$

By directly calculating (11), we find that it is small compared to the pseudo difference quotient of $\boldsymbol{\epsilon}^*$ which happens to be $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ approximately when λ approaches 1, demonstrating a smoother local behavior. Furthermore, the value remains unchanged for a duration as λ moves from about 0.5 to 1, revealing that the geometric landscape exhibits an approximate straight-line pattern with a consistent slope in the direction of $\mathbf{x}_t - \mathbf{y}_t$ around the point $\mathbf{x}_t$. In contrast, $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ has smaller difference quotient. Thus the geometry terrain along $\mathbf{x}_t - \mathbf{y}_t$ near the point $\mathbf{x}_t$ appears to be flatter.

The brief results are in Figure 2 and the detailed experiments are presented in Appendix E where we analyze the reason why the pseudo difference quotient of $\boldsymbol{\epsilon}^*$ approaches $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ when λ is closed to 1.

Jacobian. In addition to the difference quotient, the Jacobian matrix is a direct but more expensive way to reflect the local properties of a vector-valued function. Inspired by the closed form expression of $\boldsymbol{\epsilon}^*$, we can compute the Jacobian matrix of $\frac{\mathbf{x}}{\sqrt{1-\bar{\alpha}_t}} - \boldsymbol{\epsilon}^*(\mathbf{x}, t)$ for a fixed t to analyze the softmax part of $\boldsymbol{\epsilon}^*$. Similarly, we perform the same method on the trained score trying to figure out the local behavior of $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. In particular, we define

$$J_\theta^t(\mathbf{x}) \triangleq \frac{1}{\sqrt{1-\bar{\alpha}_t}}\mathbf{I} - \nabla_{\mathbf{x}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}, t), \tag{13}$$

$$J_*^t(\mathbf{x}) \triangleq \frac{1}{\sqrt{1-\bar{\alpha}_t}}\mathbf{I} - \nabla_{\mathbf{x}}\boldsymbol{\epsilon}^*(\mathbf{x}, t). \tag{14}$$

Then we calculate singular value decomposition(SVD) on $J_\theta^t(\mathbf{x})$ and $J_*^t(\mathbf{x})$ to compare their singular values and vectors. The visualization in Figure 3 shows that J_*^t has one large singular value while the singular values curve of J_θ^t is

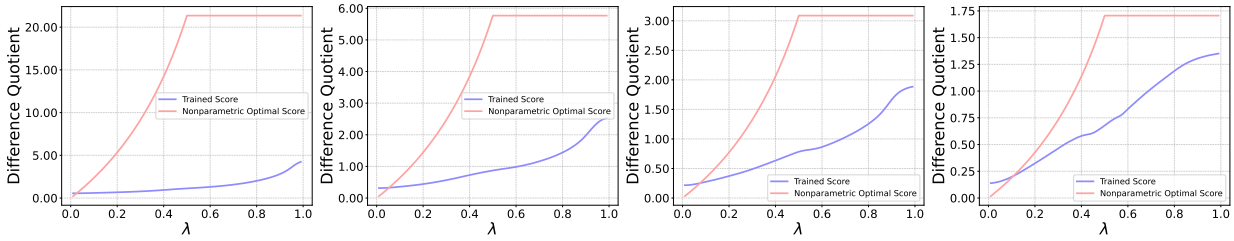
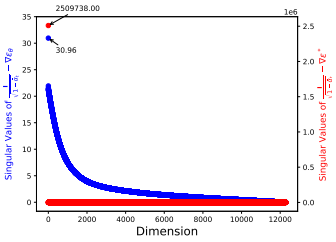
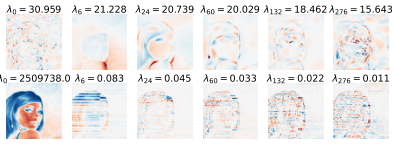


Figure 2: Results of pseudo difference quotient of trained score and nonparametric optimal score. The four figures from left to right represent the time $t = 10, 50, 100, 200$ respectively. The interpolation points $\mathbf{x}_0, \mathbf{y}_0$ are chosen randomly from the training set and each figure shares the same $\mathbf{x}_0, \mathbf{y}_0$.



(a) Singular values of $J_\theta^t(x)$ and $J_*^t(x)$ when $t = 10$.



(b) Visualization of the first dimension of the singular vectors. The first row and the second row show the singular vectors of $J_\theta^t(x)$ and $J_*^t(x)$ respectively.

Figure 3: Visualization of SVD on $J_\theta^t(x)$ and $J_*^t(x)$.

smoother. The singular vector corresponding to the biggest singular value of J_*^t presents a clear contour of the input image whereas the singular vectors of J_θ^t are fuzzier. Thus we consider J_*^t to carry most information in one single direction while J_θ^t distributes information to several different singular vector directions. See the detailed results in Appendix F.

3.3. Sampling with Mixture of ϵ_θ and ϵ^*

Since we have unraveled the discrepancies and connections between memorization and generation from both data perspective and score function perspective, we now wonder the impact of the two score functions on generation. A most direct way is to mix them up during a sampling process. That is, what will happen if $\boldsymbol{\epsilon}^*$ exclusively contributes to the reverse process within a defined interval $i \leq t \leq j$ while $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is involved for the remaining duration?

In order to explore the answer, we define

$$\boldsymbol{\epsilon}_{\boldsymbol{i}}^{\boldsymbol{j}}(\mathbf{x}, t) \triangleq \begin{cases} \boldsymbol{\epsilon}^*(\mathbf{x}, t), & i \leq t \leq j. \\ \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}, t), & \text{otherwise.} \end{cases} \tag{15}$$

and go through the DDPM reverse process with $\boldsymbol{\epsilon}_{\boldsymbol{i}}^{\boldsymbol{j}}(\mathbf{x}, t)$.

Table 1: Memorization ratios of Mixed Sampling on CIFAR10 and CelebA. A start time and end time of 0 in the first row means the memorization results of standard DDPM sampler. The third column and the fourth column present the memorization ratio of the generated 64 images of CIFAR10 and CelebA respectively.

start time i	end time j	CIFAR10 (%)	CelebA (%)
0	0	68.75	68.75
5	10	100.00	100.00
50	100	98.44	100.00
100	200	89.06	100.00
200	300	78.13	100.00
500	600	75.00	73.44
700	800	68.75	68.75

We find that the model cannot generate new images even if only a few steps are interfered by $\boldsymbol{\epsilon}^*(\mathbf{x}, t)$ when t is small enough. The detailed experiments are presented in Appendix D and one can see the sampling algorithm in Algorithm 2. We briefly show the memorization ratios of mixed sampling in Table 1, illustrating that as the mixture interval $[i, j]$ moves to the noisier side of the sampling process, the memorization ratios decrease and finally arrive at the same ratio as standard DDPM sampling. It reveals the strong ability of $\boldsymbol{\epsilon}^*$ to "redirect" the generative process to the training set.

4. Conclusion

In this study, we try to understand the discrepancy and connection between memorization and generation in diffusion models. Our exploration encompasses various aspects, starting from generated data and extending to the analysis of the scores, ultimately leading to an assessment of the sampling process. We conclude that generated data, similar to training data, are also local minima. Besides, the trained score has smoother local properties than the nonparametric optimal one, indicating potential to generalize. Finally, we mix the two scores in one DDPM sampling process to explore the impact of them on sampling.

References

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270, 2023.

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

Gu, X., Du, C., Pang, T., Li, C., Lin, M., and Wang, Y. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Kadkhodaie, Z., Guth, F., Simoncelli, E. P., and Mallat, S. Generalization in diffusion models arises from geometry-adaptive harmonic representation. *arXiv preprint arXiv:2310.02557*, 2023.

Li, P., Li, Z., Zhang, H., and Bian, J. On the generalization properties of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

Oko, K., Akiyama, S., and Suzuki, T. Diffusion models are minimax optimal distribution estimators. In *International Conference on Machine Learning*, pp. 26517–26582. PMLR, 2023.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6048–6058, 2023a.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36:47783–47803, 2023b.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. arxiv. *arXiv preprint arXiv:2106.05931*, 2021.

Wen, Y., Liu, Y., Chen, C., and Lyu, L. Detecting, explaining, and mitigating memorization in diffusion models. In *The Twelfth International Conference on Learning Representations*, 2023.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Yang, M., Zhang, C., Xu, Y., Xu, Z., Wang, H., Raj, B., and Yu, D. Usee: Unified speech enhancement and editing with conditional diffusion models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7125–7129. IEEE, 2024.

Yi, M., Sun, J., and Li, Z. On the generalization of diffusion model. *arXiv preprint arXiv:2305.14712*, 2023.

Yoon, T., Choi, J. Y., Kwon, S., and Ryu, E. K. Diffusion probabilistic models generalize when they fail to memorize. In *ICML 2023 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2023.

Zhang, H., Zhou, J., Lu, Y., Guo, M., Shen, L., and Qu, Q. The emergence of reproducibility and consistency in diffusion models. 2023.

A. Implementaion Details

In this work, we mainly do our experiments on CIFAR10 and CelebA dataset. According to previous work, small datasets and big neural networks are more likely to trigger the memorization effect which is closely related to our focus. Thus we randomly pick 10 thousand pictures(1 thousand for each class) from the CIFAR10 dataset and 7 thousand pictures from the CelebA dataset. Besides, the images in CelebA are all resized to shape of $(64, 64, 3)$. Moreover, in order to make the memorization phenomenon more pronounced, we give up all the tricks like dropout and data augmentation, which are considered beneficial to generalization.

We use the same memorization definition and metric as Carlini et al. (2023). Here we expand on the definition of "a memorized image" and the metric to evaluate memorization.

Definition A.1 (Memorization)**.** A generated image $\mathbf{x}$ is called "memorized" if there exists a image $\mathbf{x}^*$ in the training set, with

$$l(\mathbf{x}, \mathbf{x}^*) \leq \delta, \tag{16}$$

where the metric l is defined as

$$l(\mathbf{x}, \mathbf{y}) \triangleq \frac{l_2(\mathbf{x}, \mathbf{y})}{\alpha \mathbb{E}_{\mathbf{z} \in S_\mathbf{y}}[l_2(\mathbf{x}, \mathbf{z})]}, \tag{17}$$

where $l_2(\mathbf{x}, \mathbf{y}) \triangleq \sqrt{\sum_i (x_i - y_i)^2 / d}$ denotes the normalized l_2 distance, and $S_\mathbf{y}$ is the set containing the n closest samples from the training set to the sample $\mathbf{x}$. We follow the default values of $n = 50$ and $\alpha = 0.5$ in the literature. We empirically select $\delta = 0.12$ as the memorization threshold.

B. Mathematical Tools

We expand on some mathematical tools used in this work for completeness.

B.1. Difference Quotient

For a function $f(\cdot) : \mathbb{R} \to \mathbb{R}$, the first order difference quotient defined as

$$\frac{f(x) - f(y)}{x - y} \tag{18}$$

is the slope of the line connecting two points $(x, f(x)), (y, f(y))$. It is practically adopted as an approximation to the first-order derivative of $f(x)$ when x and y are close enough, making it a great criterion to determine the local smoothness of function f.

Analogously, when it comes to a vector-valued function $\mathbf{g}(\cdot) : \mathbb{R}^m \to \mathbb{R}^n$, the "pseudo difference quotient" can be defined as

$$\frac{\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\|}{\|\mathbf{x} - \mathbf{y}\|}. \tag{19}$$

From the mean value theorem(Theorem B.1), the pseudo difference quotient provides a lower bound estimation for the norm of the Jacobian matrix of $\mathbf{g}$ at a certain point $\boldsymbol{\xi} \in \{t\mathbf{x} + (1-t)\mathbf{y} | t \in (0, 1)\}$. Moreover, if $\mathbf{y} = \mathbf{x} + \delta\mathbf{d}$ where $\|\mathbf{d}\| = 1$ and $\delta \in \mathbb{R}$ has sufficiently small absolute value, the pseudo difference quotient can be considered as $\|\mathcal{J}\mathbf{g}(\mathbf{x})\|$ approximately.

Theorem B.1 (Mean Value Theorem for Vector Function)**.** *Assuming E is a convex open set in $\mathbb{R}^m$ and $\mathbf{g} : E \to \mathbb{R}^n$ is a differentiable mapping, then $\forall \mathbf{x}, \mathbf{y} \in E$, there exists $\boldsymbol{\xi} \in \{t\mathbf{x} + (1-t)\mathbf{y} | t \in (0, 1)\}$, s.t.*

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq \|\mathcal{J}\mathbf{g}(\boldsymbol{\xi})\| \|\mathbf{x} - \mathbf{y}\|. \tag{20}$$

In particular, if there exists real number M satisfying $\|\mathcal{J}\mathbf{g}(\mathbf{x})\| \leq M, \forall \mathbf{x} \in E$, then we have

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq M \|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in E. \tag{21}$$

B.2. Singular Value Decomposition

Here we show the singular value decomposition of Jacobian matrix. Suppose $\nabla_{\mathbf{x}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x},t)=\mathbf{USV}^T$ for fixed time t and input $\mathbf{x}$. Assume that the first order Taylor expansion of $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x},t)$ can approximate it, then we have

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x},t)\approx\nabla_{\mathbf{x}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x},t)\mathbf{x}=\mathbf{USV}^T\mathbf{x}. \tag{22}$$

Since all the column vectors $\mathbf{v}_i\in\mathbb{R}^n$ in matrix $\mathbf{V}$ form a standard orthogonal basis for the whole space, we have the following decomposition

$$\mathbf{x}=\sum_{i=1}^{n}\langle\mathbf{x},\mathbf{v}_i\rangle\mathbf{v}_i. \tag{23}$$

Thus $\mathbf{V}^T\mathbf{x}$ turns out to be a column vector of $\langle\mathbf{x},\mathbf{v}_i\rangle$. Then we have

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x},t)\approx\nabla_{\mathbf{x}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x},t)\mathbf{x}=\mathbf{USV}^T\mathbf{x}=\sum_k s_k(\mathbf{x})\langle\mathbf{x},\mathbf{v}_k(\mathbf{x})\rangle\mathbf{u}_k(\mathbf{x}), \tag{24}$$

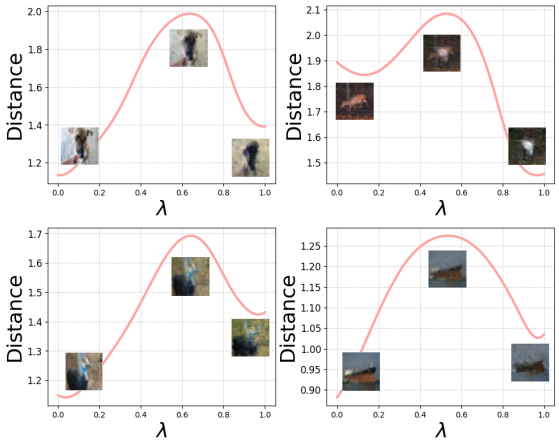
where $s_k(\mathbf{x})$ denotes the singular values of Jacobian matrix given input $\mathbf{x}$. This decomposition tells that the trained score can be interpreted as shrinkage with factors s_i along axes of a basis specified by the singular vectors $\mathbf{u}_i(\mathbf{x}),\mathbf{v}_i(\mathbf{x})$.

C. Results of $\hat{x}_\theta(\mathbf{x},t)$

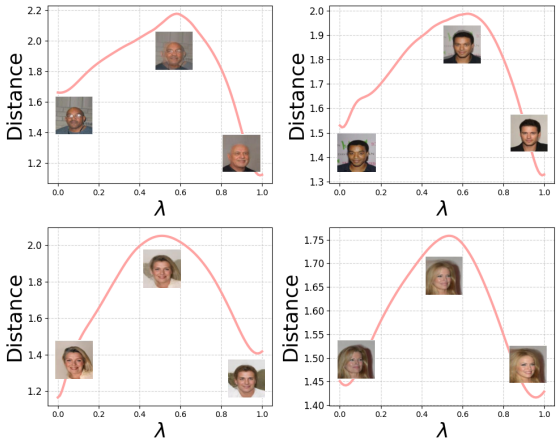
Take another look at $l_\theta^t(\lambda)$, we can further derive

$$\begin{aligned} l_\theta^t(\lambda)&=\|\hat{x}_\theta(\mathbf{z}_t(\lambda),t)-\mathbf{z}_0(\lambda)\| \\ &=\left\|\frac{\sqrt{\bar{\alpha}_t}\mathbf{z}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}-\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)}{\sqrt{\bar{\alpha}_t}}-\mathbf{z}_0\right\| \\ &=\left\|\frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t(\lambda),t)-\boldsymbol{\epsilon}\right)\right\|. \end{aligned} \tag{25}$$

If we ignore the coefficient $\frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}$, (25) is exactly the training loss function with respect to one specific data instead of taking expectations on $\mathbf{x}_0\sim q_0(\mathbf{x})$. Thus l_θ^t describes how precisely the neural network predicts the noise $\boldsymbol{\epsilon}$ given input noisy data $\mathbf{z}_t(\lambda)$.



(a) Visualization of $l_\theta^t(\lambda)$ on CIFAR10 with $t=10$.



(b) Visualization of $l_\theta^t(\lambda)$ on CelebA with $t=10$.

Figure 4: More visualization results of $l(\lambda)$ on CIFAR10 and CelebA. $\lambda=1$ refers to generated image while $\lambda=0$ refers to its nearest neighbor in the training set. We show $\mathbf{z}_0(0),\mathbf{z}_0(0.5),\mathbf{z}_0(1)$ from left to right for better visualization.

It turns out that the curve of $l_\theta^t(\lambda)$ is convex, which means there exist no local minima on the line $\{t\mathbf{x}_t+(1-t)\mathbf{y}_t,|t\in(0,1)\}$. According to (25), since local minima of $l(\lambda),\lambda\in[0,1]$ are attained at the endpoints of the interval $[0,1]$ rather than in the interior, the generated images are local minima, which are similar to training data. Moreover, we argue that the trained score does not generate new data through pixel-wise interpolation.

D. Mixed Sampling

We show both the typical DDPM sampling algorithm and the entire algorithm for mixed sampling based on DDPM sampling here.

Algorithm 1 DDPM sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Algorithm 2 Mixed Sample with DDPM

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  if $i < t < j$ **then**
    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}^*(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
  else
    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
  end if
end for
return $\mathbf{x}_0$

For the DDPM sampler, we mix trained score with nonparametric optimal score to figure out whether the injection of $\boldsymbol{\epsilon}^*$ will make some differences to the generative images. We fix random seed to exclude the effect of random noise on the results. We also pick out the nearest neighbors of generated images to determine whether they are identical.

We claim that the nonparametric optimal score has a strong ability to "redirect" the generative process to point at the nearest training sample because the softmax vector in $\boldsymbol{\epsilon}^*$ collapses rapidly to a vector with only one component is 0 and all the others are 1. Therefore, after the generative process is meddled with the nonparametric optimal score when t is relatively small(about 100), the $\mathbf{x}_t^{\leftarrow}$ contains much information of a certain training sample, with which the well-trained score is very familiar with. The trained score function is more likely to recover it rather than jump out of the high probability region to generate a brand new sample.

Figure 5 and 6 show that with the nonparametric optimal score intervening the sampling process at t small enough, DDPM sampler is able to force what would otherwise be a new image to become a replica of some image in the training set. However, when t is big, despite enlarged interval(e.g. 200 and 300 in Figure 7 and 8) the nonparametric optimal score does not necessarily lead to a nearest memorized training sample, because when t is big, both $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and $\boldsymbol{\epsilon}^*$ are noisy.

Figure 5: Results on CIFAR10. Generated images with different start time and end time. The images are presented in pairs, in which the left ones are generated images and the right ones are their nearest neighbors in the training set. Images in the top line are generated by a standard DDPM sampler. The next few lines follow the order of start time, end time as follows: $(5, 10), (50, 100), (100, 200), (200, 300), (500, 600), (700, 800)$

Figure 6: Results on CelebA. Generated images with different start time and end time. The images are presented in pairs, in which the left ones are generated images and the right ones are their nearest neighbors in the training set. Images in the top line are generated by a standard DDPM sampler. The next few lines follow the order of start time, end time as follows: $(5, 10), (50, 100), (100, 200), (200, 300), (500, 600), (700, 800)$

Figure 7: Results on CIFAR10. Generated images with different start time and end time. The images are presented in pairs, in which the left ones are generated images and the right ones are their nearest neighbors in the training set. Images in the top line are generated by a standard DDPM sampler. The next few lines follow the order of start time, end time as follows: $(200, 400), (300, 500), (400, 600), (500, 800), (600, 900), (700, 1000)$

Figure 8: Results on CelebA. Generated images with different start time and end time. The images are presented in pairs, in which the left ones are generated images and the right ones are their nearest neighbors in the training set. Images in the top line are generated by a standard DDPM sampler. The next few lines follow the order of start time, end time as follows: $(200, 400), (300, 500), (400, 600), (500, 800), (600, 900), (700, 1000)$

E. Results of Difference Quotient

The detailed analysis of difference quotient and its high-dimensional version is presented in Appendix B.1. Here we focus on the results related to our experiments.

As is stated in Section 3.2, we calculate

$$\frac{\|\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\lambda\mathbf{x}_t + (1-\lambda)\mathbf{y}_t, t)\|}{(1-\lambda)\|\mathbf{x}_t - \mathbf{y}_t\|}, \lambda \in [0, 1), \tag{26}$$

and

$$\frac{\|\boldsymbol{\epsilon}^*(\mathbf{x}_t, t) - \boldsymbol{\epsilon}^*(\lambda\mathbf{x}_t + (1-\lambda)\mathbf{y}_t, t)\|}{(1-\lambda)\|\mathbf{x}_t - \mathbf{y}_t\|}, \lambda \in [0, 1), \tag{27}$$

for fixed t and $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \mathbf{y}_t = \sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\mathbf{x}_0, \mathbf{y}_0$ are random samples from the training set. Since the pseudo difference quotient can be viewed as a function with respect to the interpolation parameter λ whenever $\mathbf{x}_t, \mathbf{y}_t, t, \mathbf{f}$ are all fixed:

$$D_{\mathbf{f}}(\cdot) : \lambda \in [0, 1) \mapsto \frac{\|\mathbf{f}(\mathbf{x}_t, t) - \mathbf{f}(\lambda\mathbf{x}_t + (1-\lambda)\mathbf{y}_t, t)\|}{(1-\lambda)\|\mathbf{x}_t - \mathbf{y}_t\|}, \tag{28}$$

we can plot the curve of both $D_{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}}(\lambda)$ and $D_{\boldsymbol{\epsilon}^*}(\lambda)$ with respect to λ. The results are shown in Figure 10.

We surprisingly find that the pseudo difference quotient of the nonparametric optimal score is extremely closed to the corresponding $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ when λ is closed to 1, whichever two samples we select from the training set. For better comparison, we show the curve of $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ in Figure 9 for reference where β_t follows linear schedule. Here we analyze the reason for this phenomenon.

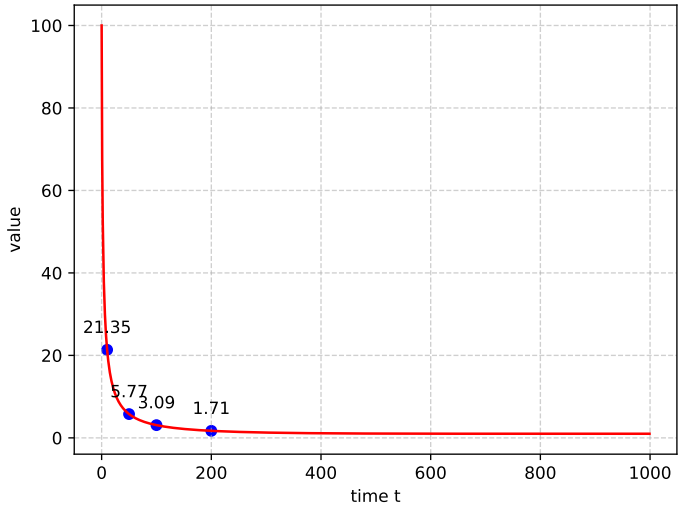


Figure 9: Curve for $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ with respect to $t \in [0, T]$

For a fixed small t, when λ approaches 1, the softmax part of $\boldsymbol{\epsilon}^*$ remains unchanged because $1-\bar{\alpha}_t$ is closed to 0 while $\left\|\mathbf{x} - \sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\right\| \approx \left\|\lambda\mathbf{x} + (1-\lambda)\mathbf{y} - \sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\right\|$. For notational convenience, we define the "inner product" of a vector and a set of vectors:

Definition E.1. Suppose a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and a serious of vectors $\{\mathbf{y_i}\}_{i=1}^n$ where $\mathbf{y_i} \in \mathbb{R}^m$. We define an operation $\otimes$ as follows

$$\mathbf{x} \otimes \{\mathbf{y_i}\}_{i=1}^n \triangleq \sum_{i=1}^n x_i\mathbf{y_i} \in \mathbb{R}^m. \tag{29}$$

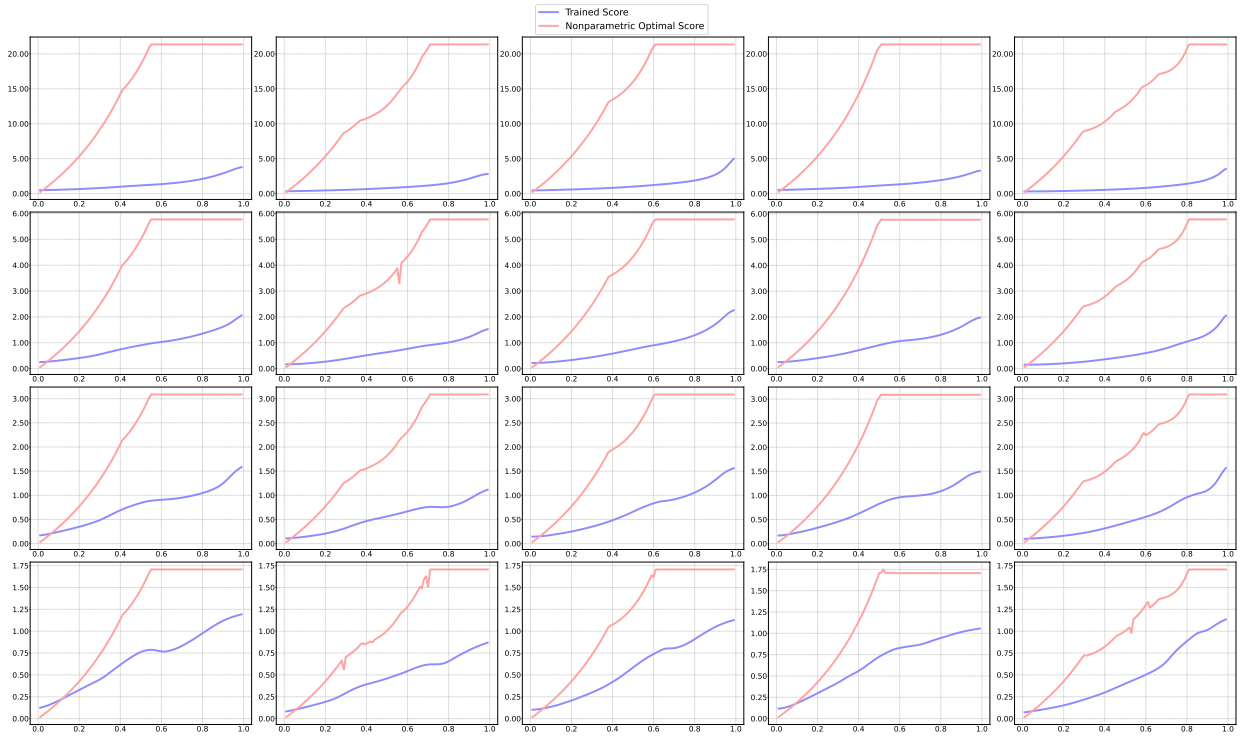


Figure 10: Results of pseudo difference quotient of trained score and nonparametric optimal score. The four rows from top to bottom represent the time $t = 10, 50, 100, 200$ respectively. Each figure in each row represents the pseudo difference quotient with respect to the interpolation parameter λ. The interpolation points $\mathbf{x}_0, \mathbf{y}_0$ are chosen randomly from the training set and each column shares the same $\mathbf{x}_0, \mathbf{y}_0$.

Based on the analysis above, we can roughly assume that

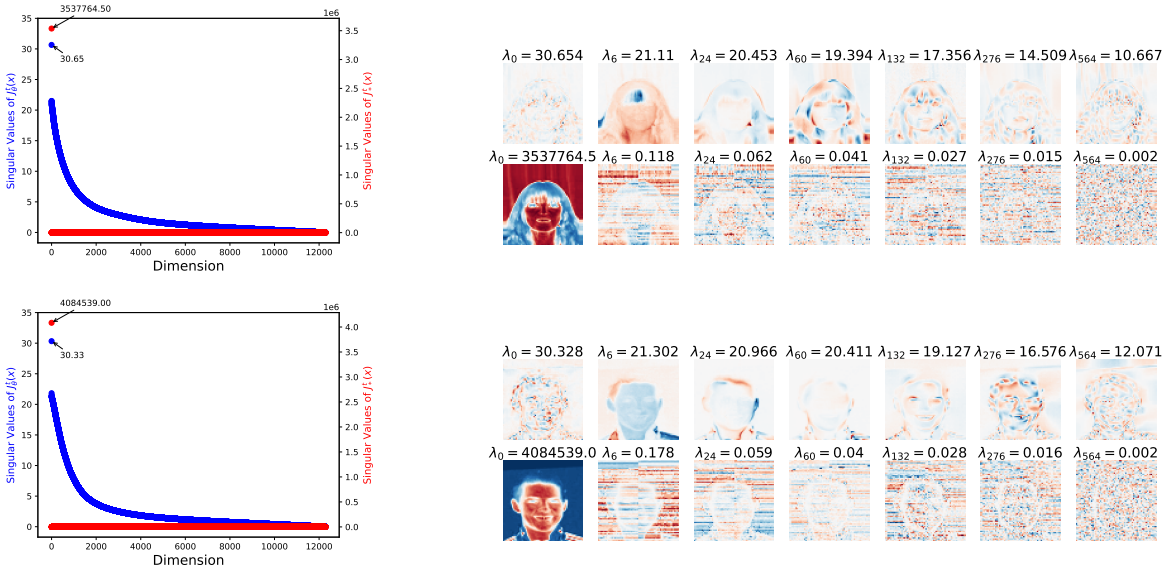
$$\begin{aligned}
&\mathcal{S}\left(-\frac{\left\|\mathbf{x}-\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\right\|^2}{2(1-\bar{\alpha}_t)}\right)\otimes\{\mathbf{x}_0^i\}_{i=1}^n \\
=&\mathcal{S}\left(-\frac{\left\|\lambda\mathbf{x}+(1-\lambda)\mathbf{y}-\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i\right\|^2}{2(1-\bar{\alpha}_t)}\right)\otimes\{\mathbf{x}_0^i\}_{i=1}^n \\
=&\mathbf{x}_0^k,
\end{aligned} \tag{30}$$

for certain positive integer k. Accordingly the only contribution to the pseudo difference quotient of $\boldsymbol{\epsilon}^*$ is $\frac{\mathbf{x}}{\sqrt{1-\bar{\alpha}_t}}$, resulting in its value being closed to $\frac{1}{\sqrt{1-\bar{\alpha}_t}}$.

In conclusion, we claim that $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is smoother than $\boldsymbol{\epsilon}^*$ at any certain training samples. In other words, despite the strong expressive power of the score neural network, the trained score is far from recovering every, or even most information carried by the training set as $\boldsymbol{\epsilon}^*$ does. That is probably the reason why the trained score has the potential of generating new samples while the nonparametric optimal score can only recover the training set.

F. Results of SVD on Jacobian Matrix

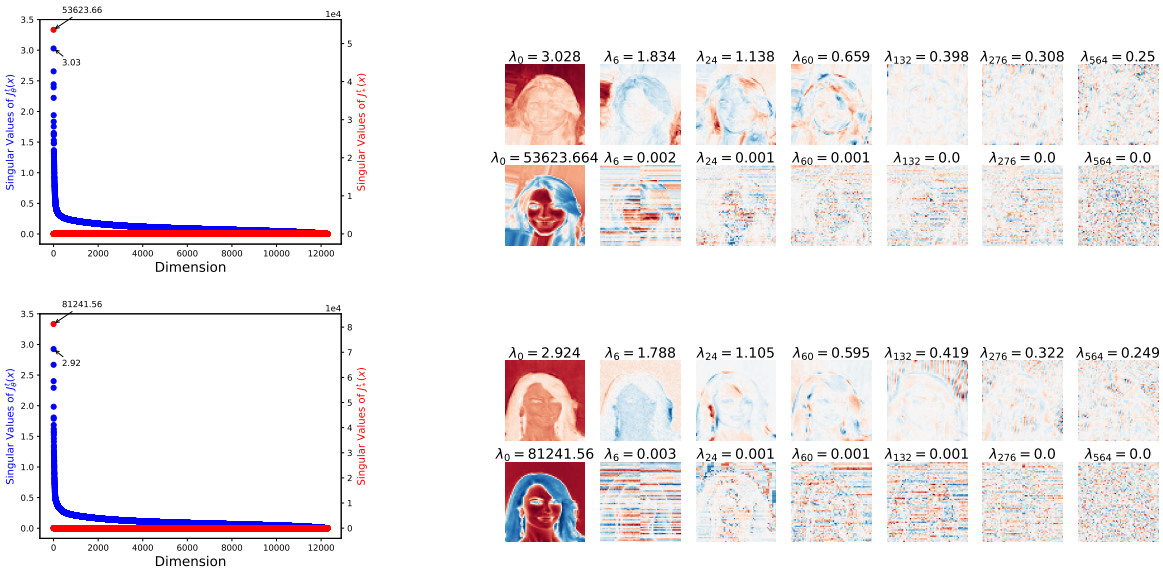
We have given detailed derivation of the application of SVD in our analysis in Appendix B.2. Now we present some visualized results of SVD on both $J_\theta^t(\mathbf{x}_t)$ and $J_*^t(\mathbf{x}_t)$ to compare the singular values and singular vectors. In addition, we repeat the same things at $t = 10, 100, 200$ so that we can better understand the local behavior of the trained score at relatively small t.



(a) Singular values of $J_\theta^t(x)$ and $J_*^t(x)$. (b) Visualization of left singular vectors. The first row shows the singular vectors of $J_\theta^t(x)$ while the second row shows the singular vectors of $J_*^t(x)$.

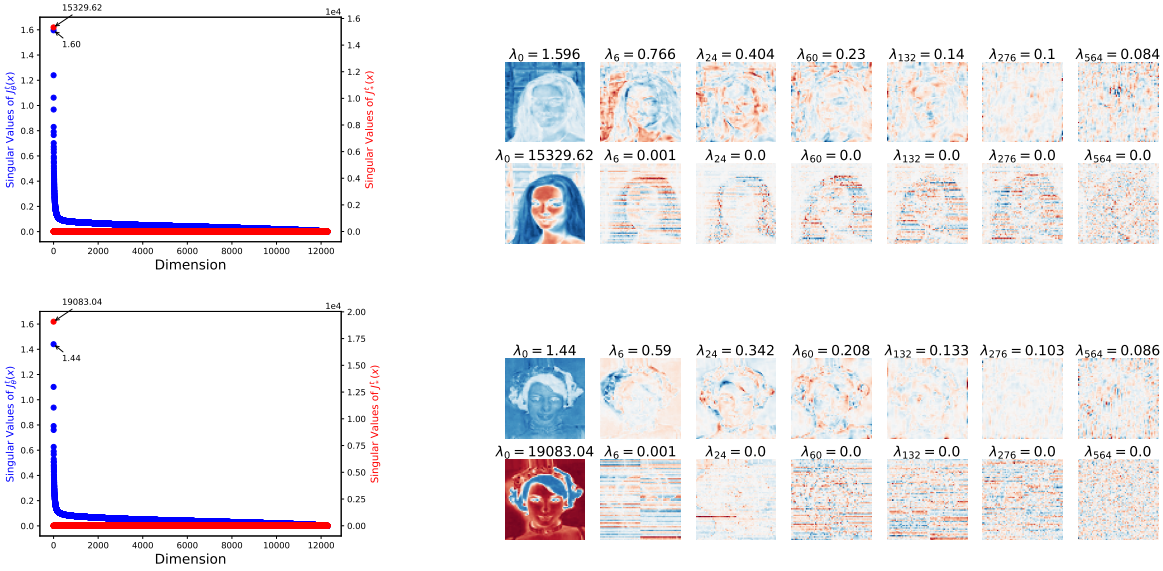
Figure 11: Singluar values of Jacobian matrix at $t = 10$. The visualized left singular vectors are only the first of three channels and use blue and red to distinguish between positive and negative pixel values.

The visualization of singular values and vectors is consistent with the results in Kadkhodaie et al. (2023) which claims that the singular vectors demonstrate oscillating patterns, adapting to the input image's geometry in both contour regions and uniformly regular areas.



(a) Singular values of $J_\theta^t(x)$ and $J_*^t(x)$. (b) Visualization of left singular vectors. The first row shows the singular vectors of $J_\theta^t(x)$ while the second row shows the singular vectors of $J_*^t(x)$.

Figure 12: Singluar values of Jacobian matrix at $t = 100$. The visualized left singular vectors are only the first of three channels and use blue and red to distinguish between positive and negative pixel values.



(a) Singular values of $J_\theta^t(x)$ and $J_*^t(x)$. (b) Visualization of left singular vectors. The first row shows the singular vectors of $J_\theta^t(x)$ while the second row shows the singular vectors of $J_*^t(x)$.

Figure 13: Singluar values of Jacobian matrix at $t = 200$. The visualized left singular vectors are only the first of three channels and use blue and red to distinguish between positive and negative pixel values.