Your Diffusion Model is Secretly a Noise Classifier and Benefits from Contrastive Training

Yunshu Wu[1] , Yingtao Luo[2], Xianghao Kong[1], Evangelos E. Papalexakis[1], Greg Ver Steeg[1]
[1]University of California Riverside, [2]Carnegie Mellon University
{ywu380,xkong016,epapalex,gregoryv}@ucr.edu, yingtaol@andrew.cmu.edu

Abstract

Diffusion models learn to denoise data and the trained denoiser is then used to generate new samples from the data distribution. In this paper, we revisit the diffusion sampling process and identify a fundamental cause of sample quality degradation: the denoiser is poorly estimated in regions that are far Outside Of the training Distribution (OOD), and the sampling process inevitably evaluates in these OOD regions. This can become problematic for all sampling methods, especially when we move to *parallel sampling* which requires us to initialize and update the entire sample trajectory of dynamics in parallel, leading to many OOD evaluations. To address this problem, we introduce a new self-supervised training objective that differentiates the levels of noise added to a sample, leading to improved OOD denoising performance. The approach is based on our observation that diffusion models implicitly define a log-likelihood ratio that distinguishes distributions with different amounts of noise, and this expression depends on denoiser performance outside the standard training distribution. We show by diverse experiments that the proposed contrastive diffusion training is effective for both sequential and parallel settings, and it improves the performance and speed of parallel samplers significantly. [1]

1 Introduction

Denoising diffusion models [29] achieve state-of-the-art performance on various unsupervised learning tasks and have intriguing theoretical connections to methods like denoising autoencoders [37], VAEs [6], stochastic differential equations [20, 34], information theory [13], and score matching [31, 32]. Diffusion models are presented with data samples corrupted by a *forward* dynamical process that progressively adds more Gaussian noise and trained to *reverse* this dynamics or denoise the corrupted samples. Samples are then generated by applying the reverse dynamics on images of pure Gaussian noise to produce high-quality samples from the target distribution.

The key to the success of diffusion models is the dynamics that gradually bridges the source and a target distribution, but it suffers from slow sampling, as sequentially simulating these dynamics can take thousands of denoising steps for one sample. Most recent works attempt to expedite the sequential dynamics by taking fewer, larger steps [30, 11, 18, 22]. However, the complexity of these samplers and the need for expensive sampling hyper-parameter grid searches tailored to specific datasets makes them difficult to generalize.

Shih et al. 2024 suggests a different approach by randomly initializing the entire path of the reverse dynamics and then updating all the steps in the path in parallel. The parallel sampling approach promises to drastically reduce wall-clock time at the cost of increased parallel computation. However, it encounters a problem that has largely gone unnoticed in the sequential sampling literature: while

---

[1]Code can be found at `https://github.com/yunshuwu/ContrastiveDiffusionLoss.git`
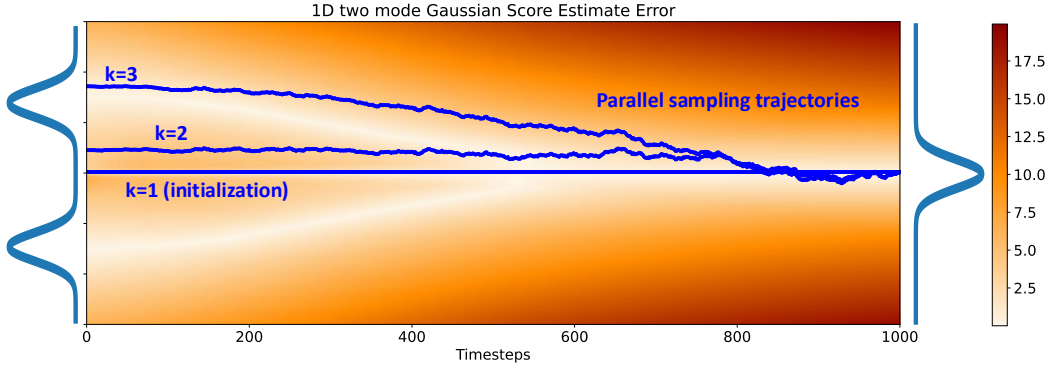
Figure 1: We plot the error in the score estimate for an 1D two mode Gaussian example where diffusion dynamics bridge between a Gaussian and a mixture (see Appendix A.3). Regions near the standard forward training data paths have lower error magnitude (light), whereas other areas have higher error magnitude (dark). While sequential samplers adhere as closely as possible to low-error regions, parallel samplers initialize and update the entire sample trajectory (blue trajectories), leading to evaluations in high-error regions. When the sampling trajectory is initialized, most are inevitably in the OOD regions and will update to the low-error regions gradually.

sequential paths sampled during generation are designed carefully to stay as close as possible to the forward paths that add noise to data, the parallel sampler often evaluates in regions far from where the score estimate (the denoiser) is trained, as illustrated in Fig. 1. The shading shows that the error of the score estimate is large in these regions, leading to poor performance for parallel samplers. Discretization errors can lead to similar issues, even for standard sequential samplers.

We propose to improve the training of diffusion models so that the error of the denoiser is reduced in OOD regions, and we hypothesize that this should significantly improve the performance for parallel samplers as they require more OOD evaluations. Our approach starts with an unexpected connection: the optimal MSE denoiser that defines the diffusion dynamics *also defines an optimal noise classifier* that distinguishes between samples with different amounts of noise. This provides a useful additional signal for training, because optimizing for the noise classification task involves evaluating the denoiser for one noise level on samples from distributions at different noise levels, while standard MSE optimization only evaluates the denoiser on samples from the matching noisy distribution. Accurate denoiser evaluation in regions that are OOD for standard diffusion training is important for robust sampling dynamics.

Contributions:

- We use the information-theoretic formulation of diffusion to draw connections between diffusion, log-likelihood ratio estimation, and classification. This reveals that optimal diffusion denoisers are also implicitly optimal classifiers for predicting the amount of noise added to an image.
- We leverage the noise classifier (via density ratio estimation [4]) interpretation to introduce a novel self-supervised loss function for regularizing diffusion model training, which we call the Contrastive Diffusion Loss (CDL). CDL provides training signal in regions that are OOD for the standard MSE diffusion loss.
- We show that CDL improves the trade-off between generation speed and sample quality, and that this advantage is consistent across different models, hyper-parameters, and sampling schemes. The improvement is especially substantial for parallel diffusion samplers [28] which rely heavily on OOD denoiser evaluations.

2 Diffusion Model Background: Optimal Denoisers are Density Estimators

The defining feature of diffusion models is a sequence of distributions that progressively add noise to the data, from which we then learn to recover the original data. The ("variance preserving" [34]) channel that mixes the signal x with Gaussian noise is defined as $x_\alpha \equiv \sqrt{\sigma(\alpha)}x + \sqrt{\sigma(-\alpha)}\epsilon$ with $\epsilon \sim \mathcal{N}(0, \mathbb{I})$, $x \sim p(x)$, where α represents the log of the Signal-to-Noise Ratio (SNR), $p(x)$ is the

unknown data distribution for $\boldsymbol{x} \in \mathbb{R}^d$, and $\sigma(\cdot)$ is the sigmoid function. We define the sequence of intermediate distributions drawn according to this channel with a subscript as $p_\alpha(\boldsymbol{x})$. By definition, we express $\lim_{\alpha \to \infty} p_\alpha(\boldsymbol{x}) = p(\boldsymbol{x})$ in this paper. Note that we use a different scaling convention for noise from [11] and [6], where the former one takes $\boldsymbol{x} + \sigma\boldsymbol{\epsilon}$ as the forward noising channel and the latter one takes $\sqrt{\alpha_t}\boldsymbol{x} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}$ as the forward noising channel. For further detailed relationships among these scaling conventions, please check App. B.3.

The minimum mean square error (MMSE) estimator $\hat{\boldsymbol{\epsilon}}$ for recovering $\boldsymbol{\epsilon}$ from the noisy channel that mixes $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$ can be derived via variational calculus and written as follows.

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_\alpha, \alpha) \equiv \mathbb{E}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}|\boldsymbol{x}_\alpha)}[\boldsymbol{\epsilon}] = \arg\min_{\tilde{\boldsymbol{\epsilon}}(\cdot, \cdot)} \mathbb{E}_{p(\boldsymbol{\epsilon})p(\boldsymbol{x})}[\|\boldsymbol{\epsilon} - \tilde{\boldsymbol{\epsilon}}(\boldsymbol{x}_\alpha, \alpha)\|_2^2]. \tag{1}$$

Sampling from the true posterior is typically intractable, but by using a neural network to approximate the solution to the regression optimization problem, we can get an approximation for $\hat{\boldsymbol{\epsilon}}$. From [13], we see that log-likelihood can be written *exactly* in terms of an expression that depends only on the MMSE solution to the Gaussian denoising problem, i.e.

$$-\log p(\boldsymbol{x}) = c + 1/2 \int_{-\infty}^{\infty} \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_\alpha, \alpha)\|_2^2]\, d\alpha. \tag{2}$$

The constant, $c = d/2 \log(2\pi e) - \frac{d}{2} \int_0^\infty d\bar{\alpha}\, \sigma(\bar{\alpha})$ does not depend on data and will play no role in our approach, as it cancels out in our derivations in Sec. 3.

3 What Your Diffusion Model is Hiding: Noise Classifiers

We now introduce our first main result, which shows that diffusion models implicitly define optimal noise classifiers. Eq. (2) expresses the probability density of the data directly in terms of the denoising function. If we apply Eq. (2) to the noisy distributions that bridge the data and a Gaussian, $p_\zeta(\boldsymbol{x})$, we can see that all mixture densities can be written in terms of the same optimal denoising function, $\hat{\boldsymbol{\epsilon}}(\cdot, \cdot)$. The complete derivation is presented in App. A.2.

$$-\log p_\zeta(\boldsymbol{x}) = c + 1/2 \int_{-\infty}^{\infty} d\alpha\, \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - b \cdot \hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_\alpha, \beta)\|_2^2] \tag{3}$$

$$\boldsymbol{x}_\alpha \equiv \sqrt{\sigma(\alpha)}\boldsymbol{x} + \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon} \tag{4}$$

$$\beta \equiv \sigma^{-1}(\sigma(\zeta)\sigma(\alpha)), \ b \equiv \sqrt{\sigma(-\alpha)/\sigma(-\beta)} \tag{5}$$

Intuitively, if we find the optimal denoising function for the data distribution, it may be hypothesized that it can denoise an already *noisy* version of the data distribution. Using Eq. 2, this directly translates into an expression for density of mixture distributions. Differences in log likelihoods lead to cancellation of constants, and these Log Likelihood Ratios (LLR) are related to the optimal classifiers [4] as we show below.

To connect LLRs with classification, consider the following generative model. We generate a random binary label $q(y = \pm 1) = 1/2$. Then, conditioned on y, we sample from some distribution $q(\boldsymbol{x}|y)$. Given samples $(\boldsymbol{x}, y) \sim q(\boldsymbol{x}, y) = q(\boldsymbol{x}|y)q(y)$, the Bayes optimal classifier is:

$$q(y|\boldsymbol{x}) = \frac{q(\boldsymbol{x}|y)q(y)}{q(\boldsymbol{x})} = \frac{q(\boldsymbol{x}|y)q(y)}{q(\boldsymbol{x}|y = 1)q(y = 1) + q(\boldsymbol{x}|y = -1)q(y = -1)}$$

$$= 1/(1 + \frac{q(\boldsymbol{x}| - y)}{q(\boldsymbol{x}|y)}) = 1/(1 + \exp(y(\log q(\boldsymbol{x}|y = -1) - \log q(\boldsymbol{x}|y = 1))))$$

$$\log q(y|\boldsymbol{x}) = -\log(1 + \exp(y \log \frac{q(\boldsymbol{x}|y = -1)}{q(\boldsymbol{x}|y = 1)})) = -\text{softplus}(y \log \frac{q(\boldsymbol{x}|y = -1)}{q(\boldsymbol{x}|y = 1)}) \tag{6}$$

In the second line, because $\forall y, q(y) = 1/2$, these constants cancel out. Then we can just expand definitions and re-arrange to write in terms of log probabilities.

Contrastive Diffusion Loss (CDL)   Our next contribution is to use the new connection between diffusion denoisers and noise classifiers to define a new training objective. We set the distributions $q(\boldsymbol{x}|y = 1)$ and $q(\boldsymbol{x}|y = -1)$ to be two distributions at different noise levels that we can write in terms of the optimal diffusion denoiser from Eq. 3. So we have $q(\boldsymbol{x}|y = 1) \equiv p(\boldsymbol{x})$, the data distribution,

and $q(\boldsymbol{x}|y = -1) \equiv p_\zeta(\boldsymbol{x})$, for some noise level, ζ. Then given a sample $(\boldsymbol{x}, y) \sim q(\boldsymbol{x}, y)$ the per-sample cross-entropy loss for the noise classifier, Eq. (6), is as follows.

$$\mathcal{L}_{CDL} = \mathbb{E}_{q(\boldsymbol{x},y)}\left[\text{softplus}(y(\log p_\zeta(\boldsymbol{x}) - \log p(\boldsymbol{x})))\right] \tag{7}$$

We can estimate both densities directly from our denoising model using Eq. (3), with the constants canceling out in the process. This loss differs significantly from the standard diffusion loss. Intuitively, to distinguish between a sample from the data distribution, $p(\boldsymbol{x})$, versus a noisy version of the data distribution, $p_\zeta(\boldsymbol{x})$, we need to evaluate denoisers on points from both distributions. In standard diffusion training, denoisers at noise level ζ are only trained on samples from $p_\zeta(x)$.

Limitations: We highlight that CDL is more expensive to compute than the standard diffusion loss, significantly increasing the total cost of diffusion model training. Implementation details appear in App. B.4 and training cost details appear in App. B.5.

Choice of noise to contrast Next, let's break the Log-Likelihood Ratio (LLR) term in Eq. (7) down to see how to choose ζ to maximize the benefit of CDL. Combining Eq. (2) and Eq. (3) we have Eq. (8), where the constant cancels out.

$$LLR = \log p_\zeta(\boldsymbol{x}) - \log p(\boldsymbol{x}) = \int_{-\infty}^{\infty} d\alpha \, \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}(\boldsymbol{z}, \alpha)\|_2^2] - \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - b\hat{\boldsymbol{\epsilon}}(\boldsymbol{z}, \beta)\|_2^2] \tag{8}$$

$$\text{with: } \boldsymbol{z} \equiv \sqrt{\sigma(\alpha)}\boldsymbol{x} + \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon}$$

Note that the input $\boldsymbol{x}$ to the LLR term may come from two different distributions, which breaks the standard synchronous denoising pair $(\boldsymbol{x}_\alpha, \alpha)$ into asynchronous. When it's from data distribution $\boldsymbol{x} \sim p(\boldsymbol{x})$, $\boldsymbol{z} = \boldsymbol{z}_\alpha$; and when it's from some noisy data distribution $\boldsymbol{x} \sim p_\zeta(\boldsymbol{x})$, $\boldsymbol{z} = \boldsymbol{z}_\beta$.

From Eq. (8) we see that $\hat{\boldsymbol{\epsilon}}(\cdot, \cdot)$ is trained on four pairs: $(\boldsymbol{z}_\alpha, \alpha)$, $(\boldsymbol{z}_\beta, \beta)$, $(\boldsymbol{z}_\beta, \alpha)$ and $(\boldsymbol{z}_\alpha, \beta)$, where $\beta \equiv \sigma^{-1}(\sigma(\alpha)\sigma(\zeta)) < \min(\alpha, \zeta)$ (Eq. 5). During standard training, only the first two pairs are trained (Eq. 1). This means that our CDL objective trains the denoiser to perform correctly even for samples from distributions that are noisier or cleaner than the specified noise level (a pair like $(\boldsymbol{z}_\beta, \alpha)$ or $(\boldsymbol{z}_\alpha, \beta)$). This can be useful for both sequential and parallel sampling settings. During sequential sampling, extra error noise added due to discretization errors can be corrected by the denoiser trained with CDL. As for parallel sampling, CDL helps with evaluations on asymmetric pairs $(\boldsymbol{z}_\beta, \alpha)$ or $(\boldsymbol{z}_\alpha, \beta)$ which we refer to OOD regions for standard diffusion loss.

In practice, diffusion training pipelines are highly tuned on popular datasets like CIFAR10 and ImageNet, so the amplitude of discretization errors during sampling is small, meaning that errors won't nudge points too far away from the true trajectory. Therefore, when evaluating CDL objective, we sample some large-valued ζs, which corresponds to classifying only small differences in noise levels. Empirically we find that $\zeta \sim Uniform[6, 15]$ or $\zeta \sim logistic[6, 15]$ performed equally good.

Denoising, sampling dynamics, and the score connection We have focused so far on denoising and density estimation, but we now want to connect this discussion to the primary use case for diffusion models and the focus of Sec. 4, *sampling*. There are many choices in how to implement sampling dynamics [11], but all of them rely on the *score function*, $\nabla_x \log p_\alpha(\boldsymbol{x})$. The score function points toward regions of space with high likelihood, and by slowly transitioning (or annealing), from the score function of a noisy distribution to one closer to the data distribution, we can build reliable sampling dynamics. To connect denoisers with sampling we must show that a denoising function, $\hat{\boldsymbol{\epsilon}}$, that is optimal according to Eq. 1 also specifies the score function.

$$\nabla_x \log p_\alpha(\boldsymbol{x}) = -\frac{\hat{\boldsymbol{\epsilon}}(\boldsymbol{x}, \alpha)}{\sqrt{\sigma(-\alpha)}} \tag{9}$$

The derivation is straightforward and is given in Appendix A.1.

4 Sequential and Parallel Sampling with Diffusion Models

Sampling dynamics are typically presented in terms of a stochastic process $\{\boldsymbol{x}_t\}_{t=1}^T$ with timestep, t, rather than in terms of log SNR, α. We will denote $\boldsymbol{x}_t \equiv \boldsymbol{x}_{\alpha(t)}, p_t(\boldsymbol{x}) \equiv p_{\alpha(t)}(\boldsymbol{x})$, to connect to our previous notation, with $\alpha(t)$ representing a monotonic relationship described in App. B.3. Note that decreasing log-SNR α corresponds to increase timestep t, since smaller log-SNR means there is more noise added to the data.

The general form of sampling dynamics is a process of slowly transitioning from samples of a simple and tractable distribution to the target distribution. Specifically, start with an isotropic Gaussian $\boldsymbol{x}_T \sim \mathcal{N}(0, \mathbb{I})$, the sampler steps through a series of intermediate distributions with noise levels $\{T, T-1, \dots, 1\}$ following the score estimates. Many works [34, 11] interpret diffusion models as stochastic differential equations (SDEs). The forward process is in the form of

$$d\boldsymbol{x}_t = \underbrace{f(\boldsymbol{x}_t, t)}_{\text{drift } s}\, dt + \underbrace{g(t)}_{\text{diffusion}}\, d\boldsymbol{w}_t, \qquad \boldsymbol{x}_0 \sim p(\boldsymbol{x}) \tag{10}$$

where $\boldsymbol{w}_t$ is the standard Wiener process/Brownian motion, f and g are drift coefficient and diffusion coefficient of $\boldsymbol{x}_t$ separately. The reverse process of Eq. 10 is then used to generate samples

$$d\boldsymbol{x}_t = \underbrace{(f(\boldsymbol{x}_t, t)\boldsymbol{x} - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}))}_{\text{drift } s}\, dt + \underbrace{g(t)}_{\text{diffusion}}\, d\boldsymbol{w}_t, \qquad \boldsymbol{x}_T \sim p(\boldsymbol{x}) \tag{11}$$

Depending on choices for f, g, we can get either a stochastic or ordinary (deterministic) differential equation. Either way, numerical differential equation solvers are used to approximate the true dynamics. The solver introduces discretization errors at each step, causing the trajectory to deviate into the OOD region where the score (or denoiser) is poorly estimated, further compounding the errors. More discretization steps reduce accumulated error and leads to better sample quality, at the expense of more sequential computation. As a result, a significant limitation of diffusion models is that they require many iterations to produce high quality samples.

Sequential Sampling The influential diffusion sampler DDPM [6] iterates over thousands of discretization steps in simulating the dynamics. Recently, many sequential sampling methods have been developed to take fewer and larger steps while introducing less error [30, 22, 11]. Specifically, Karras et al. 2022 studies the curvature shape of SDE/ODE trajectory and suggests a discretization technique where the resulting tangent of the solution trajectory always points towards the denoiser output. However, speeding up the sequential sampling sacrifices generation quality. Furthermore, the SOTA sequential samplers [18, 11, 30] require hyperparameter tuning and grid search on specific datasets, which poses challenges to the generalization of these samplers to other datasets.
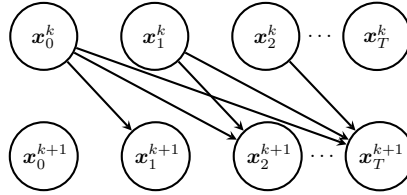

Figure 2: The computation graph of Picard iteration for parallel sampling [28]

Parallel Sampling Shih et al. 2024 explores a parallel sampling scheme, where the entire reverse process path is randomly initialized and then all steps in the path are updated in parallel. Parallel sampling is based on the method of Picard iteration, an old technique for solving ODEs through fixed-point iteration. An ODE is defined by a drift function $s(\boldsymbol{x}, t)$ and initial value $\boldsymbol{x}_0$. In the integral form, the value at time t can be written as

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + \int_0^t s(\boldsymbol{x}_u, u)\, du$$

In other words, the value at time t must be initial value plus the integral of the derivative along the path of the solution. This formula suggests a natural way of solving the ODE by starting with a guess of the solution $\{\boldsymbol{x}_t^{k+1} : 0 \le t \le 1\}$ at initial iteration $k = 0$, and iteratively refining by updating the value at every time t until convergence [2]

$$\text{(Picard Iteration)} \qquad \boldsymbol{x}_t^{k+1} = \boldsymbol{x}_0^k + \int_0^t s(\boldsymbol{x}_u^k, u)\, du \tag{12}$$

To perform Picard iterations numerically, which is shown in Fig. 2, we can write the discretized form of Eq. 12 with step size $1/T$, for $t \in [0, T]$:

$$\boldsymbol{x}_t^{k+1} = \boldsymbol{x}_0^k + \frac{1}{T}\sum_{i=0}^{t-1} s(\boldsymbol{x}_i^k, i/T) \tag{13}$$

---

[2]For detailed convergence proof, we refer to Shih et al. 2024 section 3.

We see that the expensive computations $\{s(\boldsymbol{x}_i^k, i/T) : i \in [0, T)\}$ can be performed in parallel. After some number of Picard iterations, the error difference between two iterates $\|\boldsymbol{x}_t^{k+1} - \boldsymbol{x}_t^k\|_2^2$ drops below some convergence threshold. This converged trajectory, $\boldsymbol{x}_t^*$, should be close to the sequential sampler trajectory. Looking at the example in Fig. 1, we show the trajectories of three iterations $k = 1, 2, 3$. The trajectories before convergence are consistently appearing in the regions with high score error.

5 Experiments

We sample from models fine-tuned on Contrastive Diffusion Loss (CDL) via both parallel and sequential diffusion samplers across a variety of generation tasks, including complex low-dimensional manifold 2D synthetic data and real-world image generation. Our results demonstrate that employing CDL as a regularizer in models trained with standard diffusion loss enhances density estimation and sample quality while also accelerating convergence in parallel sampling. All sampling tests are done on A6000 GPUs. We visualize CDL image generation examples in App. C.

Training configuration   Our method is architecture-agnostic. In synthetic experiments, we adopt a simple MLP architecture with positional encoding for timesteps [3], as it is one of the most versatile models in the literature. In real-world experiments, we consider the standard diffusion loss with two training configurations: DDPM by Ho et al. 2020 and EDM by Karras et al. 2022. For more details on model training, data split, and hyper-parameters, please refer to App. B.2.

Generation quality metrics   For real-world data, our intrinsic metric is Frećhet Inception Distance (FID) score [5]. The number of images we generated for FID computation follows their baseline models' FID settings, and the FID scores are computed between $5,0000$ generated images and all available real images.

For synthetic data, to measure how well the generated samples resemble samples from the ground truth distribution, we use the (unbiased) kernel estimator of the squared Maximum Mean Discrepancy (MMD), with Gaussian kernel with bandwidth set empirically as described in App. B.1.

Sampling speed metrics   We adopt the following three metrics: (1) Neural function evaluations (NFE) for all settings, i.e. how many times the denoiser is evaluated to produce a sample; (2) For the parallel setting, we report the number of parallel Picard Iterations; (3) Furthermore for the parallel setting, the wall-clock time is reported. While parallel sampling can use fewer total iterations and less wall-clock time than a sequential sampler, this may come at the cost of an increase in the total number of function evaluations. This gap is called the algorithm inefficiency. In the subsequent section, we use contrastive diffusion loss as a training regularizer for standard diffusion losses and refer to the corresponding models as CDL-regularized models.

5.1   Parallel Sampling

In the parallel setting, we use Parallel DDPM sampler [28] with 1000-step diffusion sampling. And for synthetic experiment, to reflect sampling speed by only number of Picard iteration and wall-clock time, we don't use the sliding window technique, and the 2D data is small enough to fit the whole sampler trajectory in GPU memory. While for real-world experiment, sliding window is still applied.

Synthetic Dataset   We consider the 2D Dino dataset [19], characterized by its highly nonlinear density concentrated on a low-dimensional manifold. For baselines, we employ the standard DDPM loss [6], as all standard diffusion losses similarly minimize a sum of MSE losses between the actual and estimated denoisers. Both CDL-regularized and DDPM-objective-trained models are trained with a MLP where timestep is encoded by positional encoding. We train it for 2000 epochs to ensure convergence and check the training and validation loss curve to avoid overfitting.

In Fig. 4, it is clearly demonstrated that the parallel generated samples from the CDL-regularized model is much better than the model trained only with the standard DDPM loss, especially around the chin, eyes and paws, where the manifolds are close to each other and difficult to distinguish and learn. From the MMD plot, we see that comparing to the baseline curve trained only with standard DDPM loss, the CDL-regularized curve converges faster with smaller number of Picard iterations and better

---

[3]Architecture adopted from: https://github.com/Jmkernes/Diffusion

(a) Ground-truth dino.　　(b) Using CDL loss.　　(c) Using DDPM loss.
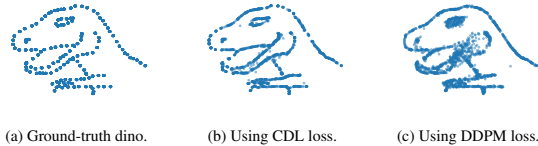
Figure 4: Parallel DDPM sampler generated Dino data. Comparing to Dino sampled from DDPM loss, CDL-loss sampled Dino has better sample quality and density estimate around hard areas.

Model Loss	MMD	#Iter	NFE	Time (ns)
DDPM	0.0031	36	14,397	1,870
CDL	**0.0012**	**27**	**13,983**	**1,368**

Table 1: Parallel DDPM sampling speed results. We generate $2,000$ samples. Here we set MMD threshold$= 0.002$, and #Iter refers to number of picard iterations till MMD threshold. Both NFE and Time are counted till parallel convergence.
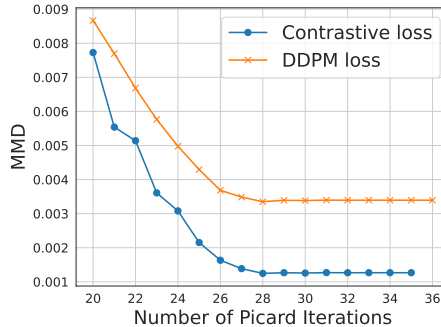
Figure 5: Parallel sampling MMD plot. We can see that the CDL-regularized model and DDPM model converge themselves at 35 and 36 Picard iterations, separately.

sample quality (lower MMD scores). Table. 1 shows the sampling speed results, from where we see that CDL-regularized model converges faster with lower final MMD and better sample quality.

Real-world Datasets　　We select "DDPM++ cont. (VP)" and "NCSN++ cont. (VE)" models by [11] trained on CIFAR-10 at 32×32, unconditional FFHQ, and unconditional AFHQv2 [15, 10, 3] as baselines, comparing to the corresponding CDL-regularized models. We adopt the pre-trained models from Ho et al. 2020[4] and Karras et al. 2022[5]. More experimental results can be found in App. B.2. As shown in Tab. 2, CDL-regularized models always outperformed baselines with respect to FID scores.

Models	CIFAR-10 at 32x32		AFHQv2 64x64	FFHQ 64x64
	unconditional	conditional	unconditional	unconditional
VP	3.24 ± 0.02	2.93 ± 0.02	2.95 ± 0.03	3.67 ± 0.04
CDL-VP	$\mathbf{2.51 \pm 0.01}$	$\mathbf{2.41 \pm 0.01}$	$\mathbf{2.91 \pm 0.02}$	$\mathbf{3.33 \pm 0.03}$
VE	3.00 ± 0.01	2.76 ± 0.01	2.98 ± 0.03	3.65 ± 0.02
CDL-VE	$\mathbf{2.38 \pm 0.01}$	$\mathbf{2.25 \pm 0.02}$	$\mathbf{2.93 \pm 0.01}$	$\mathbf{3.29 \pm 0.02}$

Table 2: Evaluating FID score (lower is better) of parallel DDPM sampler on real-world datasets using $5,0000$ samples. For reported FID scores, we run three sets of random seeds and reported the average with uncertainty.

5.2　Sequential Sampling

While CDL clearly improves parallel sampling quality and convergence speed, we also show that it improves the trade-off between generation speed and sample quality in the sequential setting. As for sequential diffusion sampling choices, we consider the DDPM sampler from Ho et al. 2020, and both the deterministic and stochastic samplers from Karras et al. 2022. To ensure fair comparisons, we adopt the original sampling hyper-parameter settings for all baselines.

Deterministic samplers　　For FID test, we follow the exact sampling settings outlined in Karras et al. 2022 for each dataset. FID scores are reported in Tab. 3, for sequential deterministic EDM samplers, CDL objective ensures that the generation quality is consistently similar or better.

In principle, increasing NFE has the potential to decrease the overall discretization errors, consequently leading to improved sample quality. However, in practice we observed an unusual behavior[6] with the Karras deterministic sampler – as NFE increases, the FID score deteriorates (Fig. 6). In contrast to EDM models, CDL-regularized models exhibit a more stable FID score. This partially resolves the deterministic sampler sensitivity while improving the quality.

Stochastic samplers　　In practice, stochastic samplers often yield superior performance compared to deterministic ones. However this is not true in Karras et al. 2022: stochastic samplers outperform

---

[4]`https://github.com/pesser/pytorch_diffusion`

[5]`https://github.com/NVlabs/edm`

[6]The same issue is also reported in `https://github.com/NVlabs/edm/issues/4`

7

Models	CIFAR-10 at 32x32		AFHQv2 64x64	FFHQ 64x64
	unconditional	conditional	unconditional	unconditional
VP	2.00 ± 0.02	1.84 ± 0.02	2.04 ± 0.00	2.38 ± 0.01
CDL-VP	$\mathbf{1.99 \pm 0.04}$	$\mathbf{1.82 \pm 0.03}$	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.29 \pm 0.02}$
VE	2.01 ± 0.01	1.81 ± 0.01	2.17 ± 0.00	2.56 ± 0.03
CDL-VE	2.01 ± 0.01	1.81 ± 0.01	$\mathbf{2.11 \pm 0.01}$	$\mathbf{2.47 \pm 0.02}$
NFE (EDM/CDL)	35	35	79	79

Table 3: Evaluating sequential deterministic EDM samplers generation quality. For reported FID scores, we run three sets of random seeds and reported the average with uncertainty.
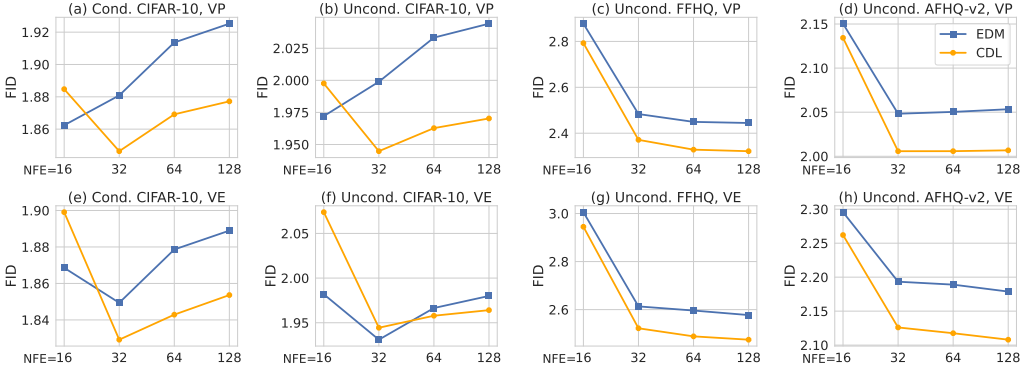

Figure 6: The FID comparison between our CDL and the baseline EDM in the deterministic sampler experiment.

deterministic ones only on challenging datasets, for simpler datasets, the introduction of stochasticity not only fails to enhance performance but exhibits image degradation issues, characterized by a loss of detail. They attribute this phenomenon to L2-trained denoisers excessively removing noise at each step (always remove more than it should), and propose to slightly increase the standard deviation (S_{noise}) [7] of newly added noise to 1.007. We argue that this approach may not totally resolve the issue and instead complicates the hyperparameter grid search process by introducing an additional parameter, S_{noise}. Also, this S_{noise} logically serves the same function as another hyperparameter γ_i, where both of them control the amount of noise to add to reach a higher noise level.

In this experiment, we conducted two stochastic sampling configurations for our baseline EDM-trained models. The first configuration, referred to as EDM-opt, operated at the EDM optimal setting with $S_{\mathrm{noise}} = 1.007$. The second, named as EDM-sub-opt, used a setting with $S_{\mathrm{noise}} = 1.00$, effectively disabling S_{noise}. As for CDL configuration, we exclusively examined the scenario with $S_{\mathrm{noise}} = 1.00$ to determine whether CDL could address the problem of excessive noise removal.

The results, as visualized in Figure 7, indicate that CDL outperforms EDM in both S_{noise} configurations. Notably, CDL not only improves upon the EDM-sub-opt configuration (dark blue line) but also surpasses the performance of the EDM-opt configuration (light blue line), even at its optimal setting. This not only demonstrates that CDL robustly provides a better sample quality, but also suggests that CDL can eliminate the need for the hyperparameter S_{noise}. This reduction enables a more efficient grid search for the optimal EDM sampling settings, potentially enhancing the practicality of using such a sampler for other applications.

6 Related Work

The generative modeling trilemma [39] seeks generative models that produce samples (i) quickly, (ii) with high quality, and (iii) good mode coverage. Diffusion models excel at the latter two but a large amount of research has attempted to address the problem of slow sampling speed. From the inception of diffusion models [29], dynamics has been at the forefront, so most work has focused on

---

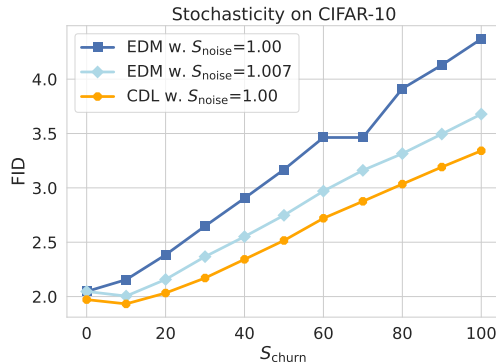[7]We refer to Karras et al. [11] for details

Figure 7: The FID comparison between our CDL and the baselines EDM in the stochastic sampler experiment on CIFAR-10. CDL's performance is strictly better for all S_{churn}, outperforming the optimal setting of EDM which inflates the standard deviation S_{churn} of the newly added noise.

the interpretation of the dynamics as a differential equation that can be sped up through accelerated numerical solvers [30, 34, 42, 9, 17]. Our approach is compatible with any of these approaches as we are sampler agnostic, seeking only to improve the input to the sampler, which is the denoiser or score function estimator, through regularization during the diffusion model training. A separate line of work instead attempts to distill a diffusion model into a faster model that achieves similar sampling quality [27, 35, 21, 38, 39]. In principle, these methods could also benefit from distilling based on a more robust base diffusion model trained with CDL.

Diffusion models admit a surprisingly diverse array of mathematical perspectives, like variational perspectives [6, 7, 12], differential equations [34, 20], and nonequilibrium thermodynamics [29]. Our approach is mostly inspired by connections between the information-theoretic perspective [13, 14] and the score matching perspective [31, 33, 32, 8, 37]. In particular, we point out that score function estimates in traditional diffusion training are sub-optimal, and the information-theoretic perspective leads to a new objective (CDL) that can improve the score estimate.

While previous diffusion models focus on log-likelihood estimation, we consider a different approach based on density ratio estimation and noise contrastive estimation [4, 23], which inspired several notable developments in machine learning [1, 25]. A few works have considered contrastive learning inspired modifications to diffusion either to enforce multimodal data relationships [16, 43], for style transfer [41], or for guidance during generation [24], but none use the diffusion model as a noise classifier to improve diffusion training as we do. Most similar to our approach are methods that use Density Ratio Estimation (DRE) to estimate a ratio between the data density and some simple noise distribution. The density ratio can be estimated by learning to contrast between data samples and samples from the noisy distribution [4, 36]. Recent work generalized the idea to consider classifying between samples along a sequence of distributions between source and target [26, 2]. Our contribution is to relate this approach to diffusion models by noting that standard diffusion models implicitly already implement the required classifiers for distinguishing distributions on the path from the data distribution to a Gaussian distribution. Concurrent work makes a similar connection but while we focus on improving diffusion models by interpreting them as noise classifiers, [40] focused on the converse perspective, improving density ratio estimation by interpreting DREs as denoisers.

7 Conclusion

In this paper, we introduced a novel connection between diffusion models and optimal noise classifiers. While this relationship has a variety of potential applications that could be explored in future work, we used the connection to propose a new self-supervised loss regularizer for diffusion models, the Contrastive Diffusion Loss (CDL). CDL reduces the error of the learned denoiser in regions that are OOD for the standard loss. We showed that CDL improves the robustness of diffusion models across all types of sampling dynamics, and leads to significant speed-ups for a promising new generation of parallel samplers.

9

Acknowledgements

Broader Impacts: This paper aims to innovate on the methodology of diffusion models. We anticipate no direct potential societal consequences of our work, as the main focus of this work is in theory and algorithm design. However, it is important to acknowledge the potential risks associated with diffusion models, as misuse can contribute to the spread of disinformation and deepfakes.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[2] Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. *arXiv preprint arXiv:2111.11010*, 2021.

[3] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[4] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

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

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

[7] 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.

[8] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[9] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

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

[11] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

[12] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *arXiv preprint arXiv:2107.00630*, 2021.

[13] Xianghao Kong, Rob Brekelmans, and Greg Ver Steeg. Information-theoretic diffusion. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2302.03792.

[14] Xianghao Kong, Ollie Liu, Han Li, Dani Yogatama, and Greg Ver Steeg. Interpretable diffusion via information decomposition. In *The Twelfth International Conference on Learning Representations*, 2024.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[16] Chaejeong Lee, Jayoung Kim, and Noseong Park. Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. *arXiv preprint arXiv:2304.12654*, 2023.

[17] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[18] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[19] Justin Matejka and George Fitzmaurice. Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 1290–1294, 2017.

[20] David McAllester. On the mathematics of diffusion models. *arXiv preprint arXiv:2301.11108*, 2023.

[21] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

[22] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[23] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[24] Yidong Ouyang, Liyan Xie, and Guang Cheng. Improving adversarial robustness by contrastive guided diffusion process. *arXiv preprint arXiv:2210.09643*, 2022.

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[26] Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

[27] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021.

[28] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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

[32] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

[33] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. *arXiv preprint arXiv:1905.07088*, 2019.

[34] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[35] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[36] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Bibliography*, page 309–326. Cambridge University Press, 2012.

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

[38] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality.

[39] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021.

[40] Shahar Yadin, Noam Elata, and Tomer Michaeli. Classification diffusion models: Revitalizing density ratio estimation. 2024.

[41] Serin Yang, Hyunmin Hwang, and Jong Chul Ye. Zero-shot contrastive loss for text-guided diffusion image style transfer. *arXiv preprint arXiv:2303.08622*, 2023.

[42] Qinsheng Zhang and Yongxin Chen. Diffusion normalizing flow. *Advances in Neural Information Processing Systems*, 34:16280–16291, 2021.

[43] Ye Zhu, Yu Wu, Kyle Olszewski, Jian Ren, Sergey Tulyakov, and Yan Yan. Discrete contrastive diffusion for cross-modal and conditional generation. *arXiv preprint arXiv:2206.07771*, 2022.

A Derivations and Proofs

A.1 Score Connection

We derive the following relation.

$$\nabla_x \log p_\alpha(\boldsymbol{x}) = -\frac{\hat{\boldsymbol{\epsilon}}(\boldsymbol{x}, \alpha)}{\sqrt{\sigma(-\alpha)}} \tag{14}$$

To keep track of intermediate random variables and their associated distributions, we will use the cumbersome but more precise information theory notation where capitals represent a random variable and lowercase represents values. Define the channel that mixes the signal, X, with Gaussian noise as $Z_\alpha \equiv \sqrt{\sigma(\alpha)}X + \sqrt{\sigma(-\alpha)}\mathcal{E}$ with $\mathcal{E} \sim \mathcal{N}(0, \mathbb{I})$ and data distribution $p(X)$, α represents the log of the Signal-to-Noise Ratio (SNR), and $\sigma(\cdot)$ is the sigmoid function. In this notation, the probability that a mixture distribution takes a value, $\boldsymbol{x}$, would be written $p(Z_\alpha = \boldsymbol{x})$.

We start by re-writing the left-hand side in new notation, and expand the definition based on the noisy channel model, using $\delta(\cdot)$ for the Dirac delta.

$$\nabla_x \log p(Z_\alpha = \boldsymbol{x}) = 1/p(Z_\alpha = \boldsymbol{x})\nabla_x p(Z_\alpha = \boldsymbol{x})$$

$$= 1/p(Z_\alpha = \boldsymbol{x})\nabla_x \int d\bar{\boldsymbol{x}}d\boldsymbol{\epsilon} \ \delta(\boldsymbol{x} - \sqrt{\sigma(\alpha)}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon})p(X = \bar{\boldsymbol{x}})p(\mathcal{E} = \boldsymbol{\epsilon})$$

$$= 1/p(Z_\alpha = \boldsymbol{x}) \int d\bar{\boldsymbol{x}}d\boldsymbol{\epsilon} \ (\nabla_x \delta(\boldsymbol{x} - \sqrt{\sigma(\alpha)}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon}))p(X = \bar{\boldsymbol{x}})p(\mathcal{E} = \boldsymbol{\epsilon})$$

$$= 1/p(Z_\alpha = \boldsymbol{x}) \int d\bar{\boldsymbol{x}}d\boldsymbol{\epsilon} \ ((-1/\sqrt{\sigma(-\alpha)})\nabla_\epsilon \delta(\boldsymbol{x} - \sqrt{\sigma(\alpha)}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon}))p(X = \bar{\boldsymbol{x}})p(\mathcal{E} = \boldsymbol{\epsilon})$$

$$= 1/p(Z_\alpha = \boldsymbol{x})\frac{1}{\sqrt{\sigma(-\alpha)}} \int d\bar{\boldsymbol{x}}d\boldsymbol{\epsilon} \ \delta(\boldsymbol{x} - \sqrt{\sigma(\alpha)}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon})p(X = \bar{\boldsymbol{x}})\nabla_\epsilon p(\mathcal{E} = \boldsymbol{\epsilon})$$

$$= -1/p(Z_\alpha = \boldsymbol{x})\frac{1}{\sqrt{\sigma(-\alpha)}} \int d\bar{\boldsymbol{x}}d\boldsymbol{\epsilon} \ \delta(\boldsymbol{x} - \sqrt{\sigma(\alpha)}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon})p(X = \bar{\boldsymbol{x}})\boldsymbol{\epsilon}p(\mathcal{E} = \boldsymbol{\epsilon})$$

$$= -1/p(Z_\alpha = \boldsymbol{x})\frac{1}{\sqrt{\sigma(-\alpha)}} \int d\boldsymbol{\epsilon} \ p(Z_\alpha = \boldsymbol{x}, \mathcal{E} = \boldsymbol{\epsilon})\boldsymbol{\epsilon}$$

$$= -\frac{1}{\sqrt{\sigma(-\alpha)}} \int d\boldsymbol{\epsilon} \ p(\mathcal{E} = \boldsymbol{\epsilon}|Z_\alpha = \boldsymbol{x})\boldsymbol{\epsilon}$$

$$= -\frac{\mathbb{E}_{\boldsymbol{\epsilon} \sim p(\mathcal{E}|Z_\alpha = \boldsymbol{x})}[\boldsymbol{\epsilon}]}{\sqrt{\sigma(-\alpha)}}$$

$$= -\frac{\hat{\boldsymbol{\epsilon}}(\boldsymbol{x}, \alpha)}{\sqrt{\sigma(-\alpha)}}$$

In the second line we expand, and in the third we just move the gradient inside the integral. In the fourth line we use the chain rule to relate the gradient over $\boldsymbol{x}$ to the gradient over $\boldsymbol{\epsilon}$ (introducing a sign flip). In the fifth line we use integration by parts to move the gradient (second sign flip). Taking the gradient of the Gaussian in the sixth line gives our third sign flip, and the factor of $\boldsymbol{\epsilon}$. We can conclude by writing the expression in terms of a conditional distribution, and relating that to the optimal denoiser in Eq. 1.

A.2 Mixture Distribution Density

In this section, we derive the expression that shows that the density of a continuum of Gaussian mixture distributions can be written in terms of the optimal denoiser, $\hat{\boldsymbol{\epsilon}}$, for the data distribution.

$$-\log p_\zeta(\boldsymbol{x}) = c + {}^1\!/_2 \int_{-\infty}^{\infty} d\bar{\alpha} \ \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - \sqrt{\frac{\sigma(-\bar{\alpha})}{\sigma(-\beta)}}\hat{\boldsymbol{\epsilon}}(\sqrt{\sigma(\bar{\alpha})}\boldsymbol{x} + \sqrt{\sigma(-\bar{\alpha})}\boldsymbol{\epsilon}, \beta)\|_2^2]$$

$$\beta \equiv \sigma^{-1}\left(\sigma(\zeta)\sigma(\bar{\alpha})\right) \tag{15}$$

13

As in the previous section, we will adopt information theory notation. If we define the optimal denoiser for the input distribution, $p_\zeta(\boldsymbol{x})$, with a subscript as $\hat{\epsilon}_\zeta(\cdot, \cdot)$, we can write the density analogously to Eq. 2.

$$-\log p_\zeta(\boldsymbol{x}) = c + \frac{1}{2} \int_{-\infty}^{\infty} d\bar{\alpha} \, \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\zeta(\sqrt{\sigma(\bar{\alpha})}\boldsymbol{x} + \sqrt{\sigma(-\bar{\alpha})}\boldsymbol{\epsilon}, \bar{\alpha})\|_2^2] \qquad (16)$$

Note that we now have to keep track of two log SNR values. One indicates how much noise is added to the new "data" distribution, the other is how much noise we add and then try to remove with our denoiser. The goal is to relate $\hat{\boldsymbol{\epsilon}}_\zeta$ to $\hat{\boldsymbol{\epsilon}}$. We can formally write down the optimal solution using the relation in Eq. 1.

$$\hat{\boldsymbol{\epsilon}}_\zeta(\boldsymbol{x}, \bar{\alpha}) = \mathbb{E}_{\boldsymbol{\epsilon} \sim p(\mathcal{E}|Z=\boldsymbol{x})}[\boldsymbol{\epsilon}]$$

Now, however, the noise channel is defined differently. The channel mixes the signal, $\bar{\boldsymbol{x}} \sim p_\zeta(\bar{X})$, with Gaussian noise, $\bar{\boldsymbol{\epsilon}} \sim \mathcal{N}(0, \mathbb{I})$, as $Z \equiv \sqrt{\sigma(\bar{\alpha})}\bar{X} + \sqrt{\sigma(-\bar{\alpha})}\bar{\mathcal{E}}$. And the noisy variable, $\bar{X} = \sqrt{\sigma(\zeta)}X + \sqrt{\sigma(-\zeta)}\mathcal{E}$, where we must be careful to distinguish the two independent sources of Gaussian noise.

We start by expanding definitions.

$$\hat{\boldsymbol{\epsilon}}_\zeta(\boldsymbol{x}_\zeta, \bar{\alpha}) = \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\bar{\boldsymbol{\epsilon}} \, p(\mathcal{E} = \bar{\boldsymbol{\epsilon}}, Z = \boldsymbol{x}_\zeta) \, \bar{\boldsymbol{\epsilon}}$$

$$= \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\bar{\boldsymbol{\epsilon}} d\bar{\boldsymbol{x}} \, \delta(\boldsymbol{x}_\zeta - \sqrt{\sigma(\bar{\alpha})}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\bar{\alpha})}\bar{\boldsymbol{\epsilon}}) \, p(\bar{X} = \bar{\boldsymbol{x}}) \, p(\bar{\mathcal{E}} = \bar{\boldsymbol{\epsilon}}) \, \bar{\boldsymbol{\epsilon}}$$

$$= \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\bar{\boldsymbol{\epsilon}} d\bar{\boldsymbol{x}} d\boldsymbol{x} d\boldsymbol{\epsilon} \, \delta(\boldsymbol{x}_\zeta - \sqrt{\sigma(\bar{\alpha})}\bar{\boldsymbol{x}} - \sqrt{\sigma(-\bar{\alpha})}\bar{\boldsymbol{\epsilon}}) \, \delta(\bar{\boldsymbol{x}} - \sqrt{\sigma(\zeta)}\boldsymbol{x} - \sqrt{\sigma(-\zeta)}\boldsymbol{\epsilon})$$
$$\cdot p(X = \boldsymbol{x}) \, p(\mathcal{E} = \boldsymbol{\epsilon}) \, p(\bar{\mathcal{E}} = \bar{\boldsymbol{\epsilon}}) \, \bar{\boldsymbol{\epsilon}}$$

$$= \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\boldsymbol{\epsilon} d\bar{\boldsymbol{\epsilon}} d\boldsymbol{x} \, \delta(\boldsymbol{x}_\zeta - \sqrt{\sigma(\bar{\alpha})}(\sqrt{\sigma(\zeta)}\boldsymbol{x} + \sqrt{\sigma(-\zeta)}\boldsymbol{\epsilon}) - \sqrt{\sigma(-\bar{\alpha})}\bar{\boldsymbol{\epsilon}})$$
$$\cdot p(X = \boldsymbol{x}) \, p(\mathcal{E} = \boldsymbol{\epsilon}) \, p(\bar{\mathcal{E}} = \bar{\boldsymbol{\epsilon}}) \, \bar{\boldsymbol{\epsilon}}$$

Now do a change of variables, a 2-d rotation with:

$$\bar{\boldsymbol{\epsilon}}' = a\bar{\boldsymbol{\epsilon}} + b\boldsymbol{\epsilon}, \boldsymbol{\epsilon}' = -b\bar{\boldsymbol{\epsilon}} + a\boldsymbol{\epsilon},$$

$$a = \sqrt{\sigma(\bar{\alpha})\sigma(-\zeta)/(1 - \sigma(\zeta)\sigma(\bar{\alpha}))}, b = \sqrt{\sigma(-\bar{\alpha})/(1 - \sigma(\zeta)\sigma(\bar{\alpha}))}.$$

This change of variables leads to the following.

$$\hat{\boldsymbol{\epsilon}}_\alpha(\boldsymbol{x}_\zeta, \bar{\alpha}) = \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\boldsymbol{\epsilon}' d\bar{\boldsymbol{\epsilon}}' d\boldsymbol{x} \, \delta(\boldsymbol{x}_\zeta - \sqrt{\sigma(\zeta)\sigma(\bar{\alpha})}\boldsymbol{x} - \sqrt{1 - \sigma(\zeta)\sigma(\bar{\alpha})}\bar{\boldsymbol{\epsilon}}')$$
$$\cdot p(X = \boldsymbol{x}) \, p(\mathcal{E}' = \boldsymbol{\epsilon}') \, p(\bar{\mathcal{E}}' = \bar{\boldsymbol{\epsilon}}') \, (b\bar{\boldsymbol{\epsilon}}' + a\boldsymbol{\epsilon}')$$

$$= b \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\boldsymbol{\epsilon}' d\bar{\boldsymbol{\epsilon}}' d\boldsymbol{x} \, \delta(\boldsymbol{x}_\zeta - \sqrt{\sigma(\zeta)\sigma(\bar{\alpha})}\boldsymbol{x} - \sqrt{1 - \sigma(\zeta)\sigma(\bar{\alpha})}\bar{\boldsymbol{\epsilon}}')$$
$$\cdot p(X = \boldsymbol{x}) \, p(\mathcal{E}' = \boldsymbol{\epsilon}') \, p(\bar{\mathcal{E}}' = \bar{\boldsymbol{\epsilon}}') \, \bar{\boldsymbol{\epsilon}}'$$

$$= b \frac{1}{p(Z = \boldsymbol{x}_\zeta)} \int d\boldsymbol{\epsilon}' d\bar{\boldsymbol{\epsilon}}' d\boldsymbol{x} \, \delta(\boldsymbol{x}_\alpha - \sqrt{\sigma(\beta)}\boldsymbol{x} - \sqrt{1 - \sigma(\beta)}\bar{\boldsymbol{\epsilon}}') \, p(X = \boldsymbol{x}) \, p(\mathcal{E}' = \boldsymbol{\epsilon}') \, p(\bar{\mathcal{E}}' = \bar{\boldsymbol{\epsilon}}') \, \bar{\boldsymbol{\epsilon}}'$$

$$\hat{\boldsymbol{\epsilon}}_\zeta(\boldsymbol{x}_\zeta, \bar{\alpha}) = b\hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_\zeta, \beta), \qquad \beta \equiv \sigma^{-1}(\sigma(\bar{\alpha})\sigma(\zeta)), b = \sqrt{\sigma(-\bar{\alpha})/(1 - \sigma(\zeta)\sigma(\bar{\alpha}))}$$

Note in the second line that the expectation of $\boldsymbol{\epsilon}'$ is zero, and we move the constant for the other term, b, outside the integral. In the third line, we define β which represents the log SNR of the two consecutive noisy channels with $\zeta, \bar{\alpha}$. Then we recognize the resulting integral as Eq. 1, the optimal denoiser for recovering samples from from the original (non-noisy) data distribution in Gaussian noise.

A.3 Main Plot 1D Two-mode Gaussian's Analytical Solution

In this section, we calculate the analytical solution to the 1D two-mode Gaussian in Fig. 1.

14

Fig. 1 is plotting the norm of difference between the ground-truth denoiser $\hat{\epsilon}_{gt}(\cdot, \cdot)$ and the estimated denoiser $\hat{\epsilon}(\cdot, \cdot)$:

$$\text{denoiser\_err}(\boldsymbol{x}, \alpha) = \|\hat{\epsilon}(\boldsymbol{x}, \alpha) - \hat{\epsilon}_{gt}(\boldsymbol{x}, \alpha)\|_2^2$$

To get this error plot, we need to analytically calculate ground-truth denoiser $\hat{\epsilon}_{gt}(\boldsymbol{x}, \alpha)$. From score connection Eq. 9, for any intermediate noisy density $\log p_\alpha(\boldsymbol{x})$, denoiser function $\hat{\epsilon}(\boldsymbol{x}, \alpha)$ can be derived from score function $\nabla_x \log p_\alpha(\boldsymbol{x})$. Therefore, ultimately what we need to calculate here is the score function of any noisy distribution $p_\alpha(\boldsymbol{x})$.

The data we used is a mixture of two Gaussians, $\mathcal{N}(\mu = -5, \mathbb{I})$ and $\mathcal{N}(\mu = 5, \mathbb{I})$, and the noise distribution is consists of data plus noise, then the noisy distribution should also be a mixture of Guassians. We just need to relate the parameters of the noisy mixture of Gaussians to the parameters of the mixture of Gaussians.

Start with one mode of the Gaussian mixture for the data. We could represent it in terms of the standard normal random variable, ϵ.

$$\boldsymbol{x}_d = \mu + \sigma\epsilon$$

Now call $\boldsymbol{x}_\alpha$ the random variable after applying a noisy channel with log-SNR α, here we present sigmoid function as $\sigma(\cdot)$.

$$\boldsymbol{x}_\alpha = \sqrt{\sigma(\alpha)}\boldsymbol{x}_d + \sqrt{\sigma(-\alpha)}\epsilon'$$

Note that we use a different ϵ' here. Now expand this, and then re-arrange.

$$\begin{aligned}\boldsymbol{x}_\alpha &= \sqrt{\sigma(\alpha)}(\mu + \sigma\epsilon) + \sqrt{\sigma(-\alpha)}\epsilon' \\ &= \sqrt{\sigma(\alpha)}\mu + \sqrt{\sigma(\alpha)}\sigma\epsilon + \sqrt{\sigma(-\alpha)}\epsilon'\end{aligned}$$

We want to represent this in a more canonical way to see what the variance and mean of this Gaussian is. Note that for two standard normal random variables, $a\epsilon + b\epsilon'$, we can represent them as a single random variable with the same variance, $\sqrt{a^2 + b^2}\epsilon''$ (reparameterization trick).

$$\boldsymbol{x}_\alpha = \sqrt{\sigma(\alpha)}\mu + \sqrt{\sigma(\alpha)\sigma + \sigma(-\alpha)}\epsilon''$$

Now we see that the noisy Gaussian (one component of a mixture) is just a modified version of the original. We have to change the mean (moving it towards zero when adding noise) and the variance.

In our example, we set $\sigma = 1$, so it simplifies further.

$$\boldsymbol{x}_\alpha = \sqrt{\sigma(\alpha)}\mu + \epsilon''$$

So the variance doesn't change, we just slowly shift the two mixtures together to the center.

Therefore, for one mode $\boldsymbol{x} \sim \mathcal{N}(\mu, I)$, the intermediate noisy log-density is.

$$\log p_\alpha(\boldsymbol{x}) = -\tfrac{1}{2}\log(2\pi\sigma^2) - \frac{(\boldsymbol{x} - \sqrt{\sigma(\alpha)}\mu)^2}{2\sigma^2}$$

Take the gradient of the log-density via torch built-in function torch.autograd.grad(log-density, samples), we have the ground-truth score function $\hat{\epsilon}_{gt}(\boldsymbol{x}, \alpha)$.

B   Implementation Details

B.1   Synthetic Experiment – Maximum Mean Discrepancy Bandwidth Choice

The Maximum Mean Discrepancy (MMD) is a statistical test used to determine if two distributions are different. It works by comparing the mean embeddings of samples drawn from two distributions in high dimensional feature space. Specifically, if the distributions are the same, the means should be close; if they are different, the means should be far apart. The embeddings are typically constructed using a feature map associated with a kernel function, and here we select the Gaussian kernel:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2})$$

15

The bandwidth parameter σ of the Gaussian kernel plays a critical role in the sensitivity and performance of MMD. The better bandwidth choice, the more effective MMD computation is. Intuitively, the bandwidth σ controls the scale at which differences between distributions are detected. A small σ makes the kernel sensitive to differences at small scales (fine details), while a large σ highlights differences at larger scales.

The choice of bandwidth is often related to the variance of the data, and the bandwidth should be on the order of the variance of the data. Through a small experiment where we calculate MMD score between our data and the standard Gaussian under various bandwidths σs, we pick the one that maximizes the MMD score.

In the synthetic 2D Dino experiment, we plot the relationship between MMD scores and bandwidths (Fig. 8), setting $\sigma = 3e - 02$.
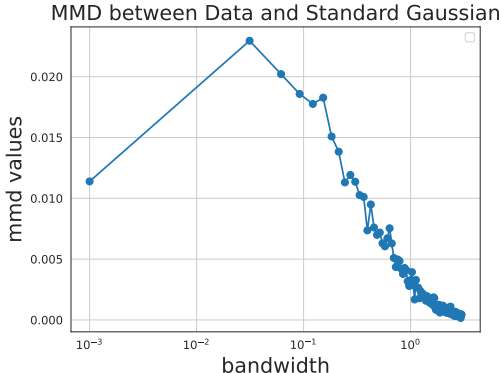

Figure 8: MMD between Dino data and the standard Gaussian. We plots the relationship between the MMD values and the bandwidth parameter used in the kernel function, and pick the bandwidth value with peak MMD score.

B.2 Details on Model Training

Model Checkpoints   We adopt two models as baselines to fine-tune with CDL: DDPM model provided by Ho et al. 2020 and EDM model provided by Karras et al. 2019.

For DDPM model, the checkpoint [8] we used is a ema one pre-trained on unconditional CIFAR-10. The reason we are not using the most frequently used checkpoint (`https://huggingface.co/google/ddpm-cifar10-32`) is that, this is not EMA checkpoint and our calculation of the FID score of this model on $50,000$ generated image via sequential DDPM sampler gives 12.43. This FID score is much higher than what reported on the original paper 3.17. Here we provide the FID scores of this non-EMA pre-trained model, results shown in Tab. 4.

	Parallel DDPM Sampler	Sequential DDPM Sampler
DDPM	10.69	12.43
CDL	**7.83**	**10.06**

Table 4: Evaluating FID score for both parallel and sequential DDPM samplers. FID scores are calculated using $5,0000$ samples.

---

[8]`https://github.com/pesser/pytorch_diffusion`

Dataset	Fine-tuning Configurations
uncond/cond CIFAR-10	`-duration=0.5 -batch=128 -lr=2e-4`
uncond AFHQ-64	`-duration=0.5 -batch=32 -lr=5e-5 -cres=1,2,2,2` `-dropout=0.25 -augment=0.15`
uncond FFHQ-64	`-batch=32 -lr=5e-5 -cres=1,2,2,2 -dropout=0.05` `-augment=0.15`

Table 5: Fine-tuning configurations for different datasets

For EDM model, in total eight checkpoints we used are "DDPM++ cont. (VP)" and "NCSN++ cont. (VE)" models pre-trained on three datasets (CIFAR-10, uncond-FFHQ, and uncond-AFHQv2 [15, 10, 3]) with two training settings (unconditional and conditional) [9] [10] [11] [12] [13] [14] [15] [16].

As for fine-tuning, we train all models with the same training setting in their original papers. For DDPM model, we train each model for 10 epochs and keep 'learning rate / batch size' ratio to be '$10^{-4}/64$', and this training is on two A6000 GPUs. For EDM model, the following table list the exact our fine-tuning configurations, which is still of the same 'learning rate/batch size' ratio. This training is using eight V100 GPUs.

More Experimental results with Parallel DDPM Sampler The previous parallel diffusion sampling paper [28] calculates FID scores by using $5,000$ generated images and another $5,000$ randomly selected real images, and to follow the same experimental setting for comparison, we further provide the FID results in Table 6.

Models	CIFAR-10 at 32x32		AFHQv2 64x64	FFHQ 64x64
	unconditional	conditional	unconditional	unconditional
DDPM	9.43	NA	NA	NA
CDL-DDPM	**9.06**	NA	NA	NA
VP	7.93 ± 0.07	7.67 ± 0.07	4.58 ± 0.07	6.26 ± 0.07
CDL-VP	$\mathbf{7.47 \pm 0.07}$	$\mathbf{7.27 \pm 0.07}$	$\mathbf{4.51 \pm 0.04}$	$\mathbf{5.89 \pm 0.07}$
VE	7.81 ± 0.07	7.59 ± 0.07	4.65 ± 0.10	6.33 ± 0.07
CDL-VE	$\mathbf{7.35 \pm 0.07}$	$\mathbf{7.19 \pm 0.07}$	$\mathbf{4.54 \pm 0.07}$	$\mathbf{5.94 \pm 0.07}$

Table 6: Evaluating FID score (lower is better) of parallel DDPM sampler on real-world datasets using $5,000$ samples. "NA" stands for "Not Applicable". For reported FID scores, we run three sets of random seeds and reported the average with uncertainty.

More Details on EDM Fine-tuning As the design choices of EDM model is very comprehensive and complicate, here we list the training noise distribution, loss weighting, network and preconditioning choices we make during CDL fine-tuning in Tab. 7.

B.3 Relationship Among Log-SNR, Timesteps, and Noise variance Sigma

To use the pre-trained models in the literature with our CDL loss, we need to translate "t", a parameter representing time in a Markov chain that progressively adds noise to data in Ho et al. 2020 and Song et al. 2020, or "σ", the variance scale of the Gaussian noise in Karras et al. 2022, to a $\log$-SNR "α".

Network and preconditioning	
Architecture of denoising function	(any)
Skip scaling $c_{skip}(\sigma)$	$\sigma_{data}^2/(\sigma^2 + \sigma_{data}^2)$
Output scaling $c_{out}(\sigma)$	$\sigma \cdot \sigma_{data}/\sqrt{\sigma_{data}^2 + \sigma^2}$
Input scaling $c_{in}(\sigma)$	$1/\sqrt{\sigma^2 + \sigma_{data}^2}$
Noise Cond. $c_{noise}(\sigma)$	$\frac{1}{4}\ln(\sigma)$
Training	
Noise distribtion	$\ln(\sigma) \sim \mathcal{N}(P_{mean}, P_{std}^2)$
Loss weighting $\lambda(\sigma)$	$(\sigma^2 + \sigma_{data}^2)/(\sigma \cdot \sigma_{data})^2$

Table 7: CDL finetune on EDM experiment – fine-tuning design choices.

Translation between Timesteps and Log-SNR　　For time-step t in DDPM and stochastic diffusion notation, we recommend readers check Kong et al. 2023 Appendix B.2 about the mapping between α and t.

Translation between Noise Variance Sigma and Log-SNR　　For variance scale of the Gaussian noise σ in EDM, referring to Eq.(7) and (8) in Karras et al. 2022, it's easy to translate the preconditioning:

$$\boldsymbol{x}_\alpha \equiv c_{in}(\sigma) \cdot (\boldsymbol{x} + \sigma\boldsymbol{\epsilon}) \tag{17}$$

$$\boldsymbol{x}_\alpha \equiv \sqrt{\sigma(\alpha)}\boldsymbol{x} + \sqrt{\sigma(-\alpha)}\boldsymbol{\epsilon} \equiv \sqrt{\sigma(\alpha)}\left(\boldsymbol{x} + \sqrt{\frac{\sigma(-\alpha)}{\sigma(\alpha)}}\boldsymbol{\epsilon}\right) \tag{18}$$

From Eq. 18 and Eq. 17, we see that $\sigma \equiv \sqrt{\frac{\sigma(-\alpha)}{\sigma(\alpha)}}$, therefore, the relationship between α and σ should be:

$$\sigma \equiv \exp(-\alpha/2), \ \alpha \equiv -2\ln(\sigma)$$

B.4 Contrastive Loss Implementation

To implement contrastive loss, we follow the definition in Sec. 3. First, we generate a random binary label y. Next, conditioned on y, we sample from either data distribution $p(\boldsymbol{x})$ or the noisy data distribution $p_\zeta(\boldsymbol{x})$. We calculate the point-wise log-likelihood, then the contrastive loss in Eq. 7.

Algorithm 1 Contrastive Diffusion Loss – Training

1: **repeat**
2: 　　$x_0 \sim p(x_0)$
3: 　　$\zeta \sim \text{Uniform}(6, \ldots, 15)$
4: 　　# uniformly sample from $p(x)$ or $p_\zeta(x)$
5: 　　**if** rand-prob < 0.5 **then**
6: 　　　　$x = x_0$
7: 　　　　$y = 1$
8: 　　**else**
9: 　　　　$x = \text{generate\_mixture}(x_0, \zeta)$
10: 　　　　$y = -1$
11: 　　**end if**
12: 　　# calculate negative log-likelihood of $p(x), p_\zeta(x)$
13: 　　log_px = -nll(x)
14: 　　log_px_zeta = -nll(x, ζ)
15: 　　cdl_loss = softmax($y\cdot$ (log_px_zeta - log_px))
16: **until** converged

B.5 Training cost

As we mentioned, CDL training is more expensive to compute than the standard diffusion loss, and here we analysis and give the reason.

According to Kong et al. 2023, we can write the pointwise standard diffusion loss function as Eq. 19, and therefore the standard diffusion loss is as Eq. 20.

$$nll(\boldsymbol{x}) = -\log p(\boldsymbol{x}) = c + \frac{1}{2} \int_{-\infty}^{\infty} \mathbb{E}_{p(\boldsymbol{\epsilon})}[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_\alpha, \alpha)\|_2^2] \, d\alpha. \tag{19}$$

$$nll = \mathbb{E}_{p(\boldsymbol{x})}[-\log p(\boldsymbol{x})] = c + \frac{1}{2} \int_{-\infty}^{\infty} \mathbb{E}_{p(\boldsymbol{\epsilon}) \ p(\boldsymbol{x})}[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_\alpha, \alpha)\|_2^2] \, d\alpha. \tag{20}$$

To train the standard diffusion loss, we simply need to optimize Eq. 20 by all the training data. However, to train the contrastive diffusion loss, we need to estimate $nll(x)$ and $nll(x + \zeta)$ term in Algo. 1, and there we estimate Eq. 19 by duplicating a single data point $\boldsymbol{x}$ for $N = 100$ times and calculate Eq. 20. This pointwise NLL estimation $nll(x)$ demands $N = 100$ times more computational resources compared to the standard diffusion loss.

In principle, the contrastive loss Algo. 1 should be executed for the entire training dataset. However, due to the high computational cost, we optimize only one data point per batch instead of utilizing all the training data.

C   Samples Visualization

We provide visualization of the images generated from pre-trained models fine-tuned via CDL loss.
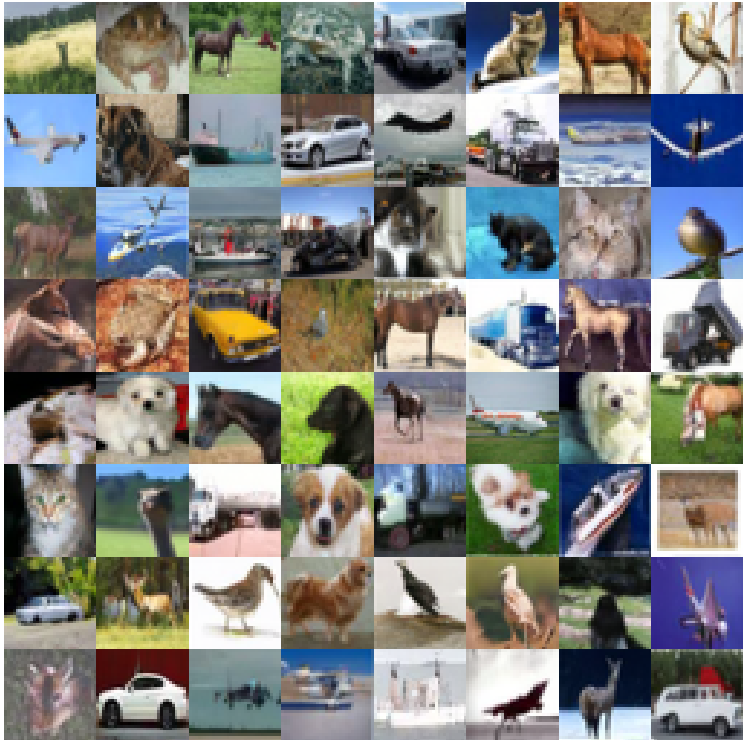

Figure 9: The CDL-loss fine-tuned EDM checkpoint generated examples from Conditional CIFAR-10, via parallel DDPM sampler.

Figure 10: The CDL-loss fine-tuned EDM checkpoint generated examples from Unconditional CIFAR-10, via parallel DDPM sampler.

Figure 11: The CDL-loss fine-tuned EDM checkpoint generated examples from Unconditional AFHQ, via parallel DDPM sampler.
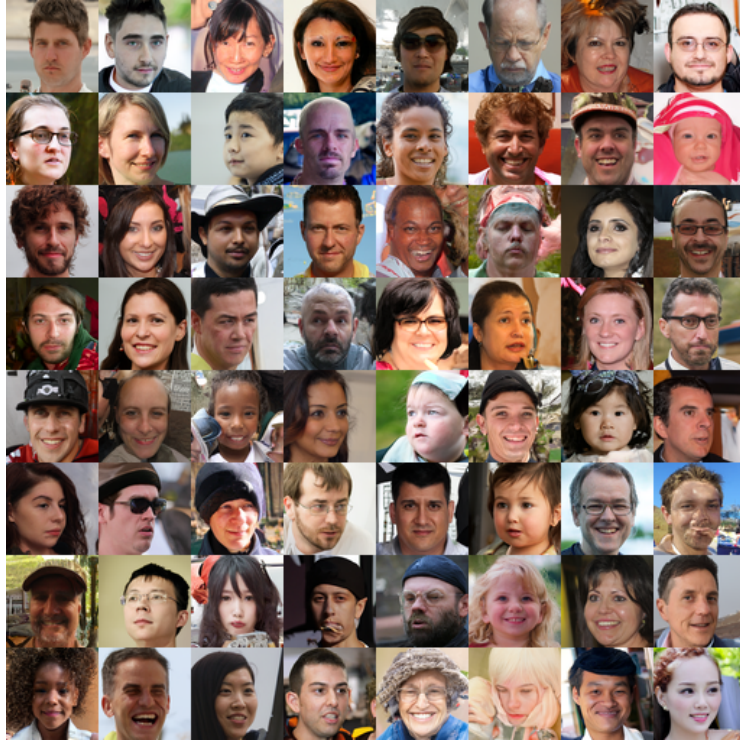
Figure 12: The CDL-loss fine-tuned EDM checkpoint generated examples from Unconditional FFHQ, via parallel DDPM sampler.
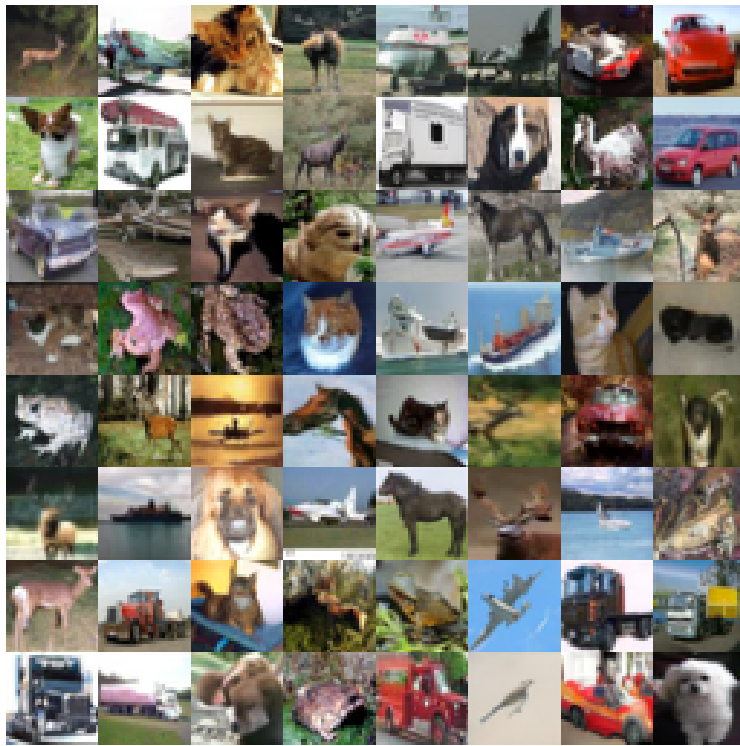

Figure 13: The CDL-loss fine-tuned EDM checkpoint generated examples from Conditional CIFAR-10, via sequential EDM sampler.

Figure 14: The CDL-loss fine-tuned EDM checkpoint generated examples from Unconditional CIFAR-10, via sequential EDM sampler.

Figure 15: The CDL-loss fine-tuned EDM checkpoint generated examples from Unconditional AFHQ, via sequential EDM sampler.

Figure 16: The CDL-loss fine-tuned EDM checkpoint generated examples from Unconditional FFHQ, via sequential EDM sampler.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We explain the proposed CDL speeds up parallel sampling convergence and improves generation quality. For sequential sampling, it improves the trade-off between sampling quality and gerneration speed. We provide the claims in both abstract and introduction, and justify them in the Results section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main limitation to our approach is the increased training cost. We explicitly highlight limitations in the methods development in Sec. 3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Every theoretical result is supported by a complete proof and assumptions as well, if applicable.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide anonymous code link and full description of training configuration.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

    Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

    Answer: [Yes]

    Justification: We provide anonymous code link and full description of training configuration.

    Guidelines:

    - The answer NA means that paper does not include experiments requiring code.
    - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
    - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
    - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
    - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
    - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
    - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
    - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

    Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

    Answer: [Yes]

    Justification: We provide all detailds about training and testing details in both experimental section and also in appendix. Since we test on benchmark, our data split are rigously the same as all other baseline settings.

    Guidelines:

    - The answer NA means that the paper does not include experiments.
    - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
    - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

    Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

    Answer: [Yes]

    Justification: We report error bars wherever possible. One set of error bars for Table 2 is missing due to lack of computing resources, but will be added in the final version.

    Guidelines:

    - The answer NA means that the paper does not include experiments.
    - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

    Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

    Answer: [Yes]

    Justification: We provide sufficient information on the computation resources, e.g. GPU information, to reporduce the experiments.

    Guidelines:

    - The answer NA means that the paper does not include experiments.
    - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
    - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
    - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

    Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

    Answer: [Yes]

    Justification: We do not use human subjects and only used standard datasets for experiments. We acknowledge the potential harms of diffusion models in the broader impacts statement, and do not anticipate that our methodological contributions will alter known issues. We also strive for transparency wherever possible, including full, anonymized code with our submission.

    Guidelines:

    - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
    - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
    - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: While we do not expect a direct societal impact from our methodological improvements, we added a broader impacts statement that acknowledges the potential impact of diffusion models.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We used existing standard datasets and public, pretrained models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We give proper attribution for the open license code, data, and models used throughout the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are provided.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing or human subjects research is included.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.