DIFFRATIO: Training One-Step Diffusion Models Without Teacher Supervision

Wenlin Chen[*]
Bosch Center for AI

Mingtian Zhang[*]
University College London

Jiajun He[*]
University of Cambridge

Zijing Ou
Imperial College London

José Miguel Hernández-Lobato
University of Cambridge

Bernhard Schölkopf
Max Planck Institute for Intelligent Systems

David Barber
University College London

Abstract

Score-based distillation methods train one-step diffusion models in two stages: they first train a teacher score model, then distill it into a one-step student model. However, this distillation process can introduce bias from two sources: errors in the teacher score model and student score estimate. We propose **DIFFRATIO**, a new approach for training one-step diffusion models without teacher supervision based on a density-ratio formulation that connects score-based distribution matching and GAN-style adversarial training. Instead of using a teacher score model to provide training targets, DiffRatio directly learns a log density ratio between the student and data distributions across diffusion time steps. As a pre-trained score model is only used to initialize the one-step generator and does not supervise training, this design simplifies the training pipeline, mitigates gradient estimation bias, and reduces the size of auxiliary networks. The learned density ratio can be used as a verifier, enabling a principled inference-time parallel scaling method that further improves sample quality without external rewards or extra sequential computation. DiffRatio achieves strong one-step generation results on CIFAR-10 and ImageNet (64×64 and 512×512), outperforming most teacher-supervised distillation methods.

1 Introduction

Diffusion models [51, 21, 56] have achieved remarkable success in modeling complex real-world data across a wide range of domains [47, 31, 44, 22, 32]. Typically, diffusion models consist of two processes: a forward noising process, which gradually perturbs data into a known noise prior (e.g., Gaussian), and a reverse denoising process, which learns to invert the forward corruption process to generate realistic data samples from noise. Formally, the forward process is defined over T time steps as a Markov chain of Gaussian transitions, while the reverse process is parameterized by neural networks that predict the denoising distribution given the noisy samples. In classic diffusion models with Gaussian denoising distributions, generating high-quality data samples typically requires hundreds or thousands of sampling steps, resulting in significant sampling inefficiency due to the need for $T \gg 1$ NFEs (number of function evaluations) [21, 41]. To address this weakness, various acceleration methods have been proposed to reduce NFEs during sampling. One class of such approaches leverages advanced numerical solvers for differential equations, enabling continuous-time approximations of the diffusion process [53, 33, 35]. Another line of work improves the

[*]Equal contribution. Correspondence to Wenlin Chen <wenlin.chen@cn.bosch.com>, Mingtian Zhang <m.zhang@cs.ucl.ac.uk>, and Jiajun He <jh2383@cam.ac.uk>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

flexibility of the posterior distribution in the denoising process, either by estimating a more accurate covariance for the Gaussian distribution [41, 5, 4, 43] or by adopting flexible non-Gaussian denoising distributions [8, 62, 66]. While these techniques can dramatically reduce NFEs from $\sim 10^3$ to around 10–20, they still fall short of achieving high-quality generation within 5 steps.

Recently, distillation-based methods have emerged as a powerful direction for training diffusion models, enabling high-quality *one-step* generation [73]. These methods fall into two categories. Trajectory-based distillation methods [49, 6, 55, 20, 27, 13, 30, 14, 7] aim to approximate the full sampling trajectory by training a student model to amortize multiple intermediate steps. These methods are motivated by accelerated solvers and typically perform joint training of the full diffusion model and the distillation process. Score-based distillation approaches [36, 65, 50, 63, 73] distill the full denoising process of the pre-trained teacher diffusion model into a one-step latent variable model. This distillation process typically involves minimizing a divergence between the student and teacher models based on their respective score estimations [44, 61] and typically follow a two-stage pipeline: (1) *pre-training stage*: training a teacher score model via denoising score matching on the data distribution, and (2) *distillation stage*: using the pre-trained teacher score to supervise the training of a one-step student model. However, the distillation-stage gradient estimator suffers from two sources of bias: (1) imperfect teacher score supervision due to score estimation error from pre-training, and (2) student score estimation error during distillation. These biases can accumulate and thus significantly degrade the quality of the resulting one-step model.

We propose **DIFFRATIO**, a new approach for training one-step diffusion models by directly estimating the *score difference* between the student and data distributions through the gradient of a learned log-density-ratio network. As a pre-trained score model is only used to initialize the one-step generator and does not supervise training, this approach mitigates gradient estimation bias, simplifies the training pipeline, and reduces the size of auxiliary networks. The learned density ratio can further act as a verifier, enabling a *principled inference-time parallel scaling scheme* that improves generation quality without external rewards or extra sequential computation. DiffRatio achieves state-of-the-art one-step generation performance across standard image generation benchmarks. Moreover, we find that initializing the student model with the teacher's weights remains critical since the teacher diffusion model provides a rich set of feature representations across different noise levels.

2 Background

2.1 Diffusion Models

Diffusion models [51, 21, 56] define a generative framework that maps a Gaussian prior $p(x_T)$ to the data distribution $p(x_0)$ through a forward noising process and a learned reverse denoising process. The forward process progressively adds Gaussian noise to the data

$$q(x_{0:T}) = p_d(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{1}$$

with transition kernels defined as $q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$, where $\beta_t \in (0,1)$ is a pre-specified variance. The skip conditional distribution at time t can be written as: $q(x_t|x_0) = \mathcal{N}(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$, where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. As $T \to \infty$, the final state x_T approximates a standard normal distribution, i.e., $q(x_T) \to \mathcal{N}(0, I)$. The generative process aims to reverse this trajectory. Starting from noise $x_T \sim p(x_T) = \mathcal{N}(0, I)$, the model learns a reverse process to sequentially denoise and reconstruct data samples. Since the true reverse conditional $q(x_{t-1}|x_t)$ is intractable, a common method is to approximate it with a variational Gaussian distribution:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_{t-1}(x_t;\theta), \Sigma_{t-1}(x_t;\theta)), \tag{2}$$

where the mean function is learned from data and the covariance function can be either learned [41, 43, 4] or chosen to be a fixed value [5, 21]. From a score-based perspective, the denoising mean $\mu_{t-1}(x_t;\theta)$ can be written in terms of the (time-dependent) score of the forward marginal $q(x_t)$, i.e., $\nabla_{x_t} \log q(x_t)$. In practice, we learn score networks using denoising score matching (DSM) [58, 56]. Specifically, for any distribution $q(x_0)$ and its noised marginal $q(x_t) = \int q(x_t|x_0)\, q(x_0)\, dx_0$, DSM trains a score network $s_\theta(x_t, t)$ to estimate $\nabla_{x_t} \log q(x_t)$ by minimizing

$$\min_\theta \; \mathbb{E}_{q(x_0)\, q(x_t|x_0)} \left[\| s_\theta(x_t, t) - \nabla_{x_t} \log q(x_t|x_0) \|_2^2 \right]. \tag{3}$$

Given an estimated score $s_\theta(x_t, t) \approx \nabla_{x_t} \log q(x_t)$, Tweedie's Lemma [11, 46] yields the corresponding reverse mean:

$$\mu_{t-1}(x_t; \theta) = \frac{x_t + \beta_t \, s_\theta(x_t, t)}{\sqrt{1-\beta_t}}, \tag{4}$$

which establishes a connection between the denoising distributional perspective and the score estimation perspective of diffusion models. In the following section, we introduce score-based distillation methods through the lens of divergence minimization.

2.2 Score-Based Diffusion Distillation

Score-based distillation methods aim to distill a teacher diffusion model p_θ (pre-trained on the true data distribution $p_d(x_0)$) into a one-step implicit generative model [16, 23, 68]:

$$q_\theta(x_0) = \int \delta(x_0 - g_\theta(z)) p(z) dz, \tag{5}$$

where $\delta(\cdot)$ is the Dirac delta function, $p(z)$ is a standard Gaussian prior for the latent variable z, and $g_\theta : \mathcal{Z} \to \mathcal{X}$ is a deterministic neural network that generates data x from the latent variable z in one step. We want to minimize the divergence between this one-step model and the true data distribution. However, when the function $g_\theta(\cdot)$ is not bijective, the model distribution q_θ is not absolutely continuous w.r.t. the Lebesgue measure. As a result, the corresponding density function may not be well-defined, and consequently, the commonly used f-divergence such as KL divergence between $q_\theta(x_0)$ and the data distribution $p_d(x_0)$ may also be ill-defined [3, 68].

Inspired by diffusion models, one can use a set of (scaled) Gaussian convolution kernels $\mathcal{K} = \{k_1, \cdots, k_T\}$ with $k_t(x_t|x_0) = \mathcal{N}(x_t|\alpha_t x_0, \sigma_t^2 I)$ to define the Diffusive KL divergence (DiKL) between the model density $q_\theta(x_0)$ and target distribution $p_d(x_0)$:

$$\mathrm{DiKL}_{\mathcal{K}}(q_\theta||p_d) \equiv \sum_{t=1}^{T} w(t) \mathrm{KL}(q_\theta(x_t)||p_d(x_t)), \tag{6}$$

where $w(t)$ is a positive scalar weighting function that sums to one, and $q_\theta(x_t)$ and $p_d(x_t)$ are noisy model density and noisy target density, respectively, defined by

$$q_\theta(x_t) = \int k(x_t|x_0) q_\theta(x_0) dx_0, \quad p_d(x_t) = \int k(x_t|x_0) p_d(x_0) dx_0. \tag{7}$$

In this case, the distributions $q_\theta(x_t)$ and $p_d(x_t)$ are always absolutely continuous, and thus the KL divergence between them is always well-defined. For a single Gaussian kernel, the divergence was known as *Spread KL divergence* [68, 67]. It is straightforward to show that it is a valid divergence, i.e., $\mathrm{DiKL}_{\mathcal{K}}(q_\theta||p_d) = 0 \Leftrightarrow q_\theta = p_d$; see [68] for a proof. In addition to the diffusion distillation [36, 63, 59], this divergence has successfully been used in 3D generative models [44, 61] and training neural samplers [19]. For a single kernel k_t, the DiKL term admits the following pathwise gradient (see Appendix A for a derivation):

$$\nabla_\theta \mathrm{DiKL}_t(q_\theta(x_0)||p_d(x_0)) = \int q_\theta(x_t) \Big(\underbrace{\nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)}_{\text{score difference at time } t} \Big) \frac{\partial x_t}{\partial \theta} dx_t. \tag{8}$$

However, neither the noisy model score $\nabla_{x_t} \log q_\theta(x_t)$ nor the noisy target score $\nabla_{x_t} \log p_d(x_t)$ are directly accessible. In the distillation setting, the noisy target score is provided by a pre-trained diffusion model. Specifically, a score network $s_{\psi_1}^{p_d}(x_t, t)$ is learned to approximate $\nabla_{x_t} \log p_d(x_t)$ by minimizing the DSM loss w.r.t. ψ_1 at time t:

$$\min_{\psi_1} \mathbb{E}_{p_d(x_0)k(x_t|x_0)} \left[\| s_{\psi_1}^{p_d}(x_t, t) - \nabla_{x_t} \log k(x_t|x_0) \|_2^2 \right]. \tag{9}$$

Regarding the noisy model score $\nabla_{x_t} \log q_\theta(x_t)$, since samples from q_θ can be efficiently obtained, we can approximate the score with another score network $s_{\psi_2}^{q_\theta}(x_t, t) \approx \nabla_{x_t} \log q_\theta(x_t)$ by minimizing the DSM loss w.r.t. ψ_2:

$$\min_{\psi_2} \mathbb{E}_{q_\theta(x_0)k(x_t|x_0)} \left[\| s_{\psi_2}^{q_\theta}(x_t, t) - \nabla_{x_t} \log k(x_t|x_0) \|_2^2 \right]. \tag{10}$$

Thus, the gradient of DiKL w.r.t. the parameters θ of the student model can be estimated as follows, a method known as Variational Score Distillation (VSD) [44, 61, 36]:

$$\nabla_\theta \mathrm{DiKL}(q_\theta(x_0)||p_d(x_0)) \approx \sum_{t=1}^{T} w(t) \int q_\theta(x_t)\big(s^{q_\theta}_{\psi_2}(x_t,t) - s^{p_d}_{\psi_1}(x_t,t)\big)\frac{\partial x_t}{\partial \theta} dx_t. \tag{11}$$

Unlike the teacher score function which remains fixed after pre-training, the noisy model score $\nabla_{x_t} \log q_\theta(x_t)$ dynamically changes as we update the student model's parameters θ during training. Therefore, the score network $s^{q_\theta}_{\psi_2}(x_t,t) \approx \nabla_{x_t} \log q_\theta(x_t)$ needs to be updated every time we update the student model, which results in an interleaved training procedure as detailed in Algorithm 2.

Limitations of the VSD. Despite the initial success of VSD in diffusion distillation, this two-stage pipeline has inherent limitations. In the first stage, the teacher score model $s^{p_d}_{\psi_1}(x_t,t)$ is imperfect due to limited network capacity and optimization. In the second stage, the distillation process solely relies on the pre-trained teacher score for supervision. Therefore, the gradient estimation in VSD (Eq. 11) has biases from *two sources*: (1) the teacher's score estimation error, and (2) the student's score estimation error. These biases can accumulate and thus adversely affect the quality of the distilled student model. In the next section, we propose a new method that directly estimates the score difference $\nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)$ with a single lightweight network, which reduces the gradient estimation bias and improves generation quality.

3 Training One-Step Diffusion Models with DiffRatio

From Eq. 8, we observe that the DiKL gradient depends on the difference between two score functions, which can be expressed as the gradient of a log density ratio:

$$\nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t) = \nabla_{x_t} \log \frac{q_\theta(x_t)}{p_d(x_t)}. \tag{12}$$

Therefore, instead of estimating the two score functions separately, we can directly learn the density ratio $\frac{q_\theta(x_t)}{p_d(x_t)}$ and then take the gradient of its logarithm to estimate the score difference. We thus refer to this approach as **DIFFRATIO**, which we introduce below.

3.1 Diffusive Density Ratio Gradient Estimation

We propose to use class ratio estimation [57, 45, 18, 69] to estimate the density ratio across all noise levels. We first denote distributions $q_\theta(x_t)$ and $p_d(x_t)$ as two conditional distributions $m(x_t|y=0)$ and $m(x_t|y=1)$, respectively, where $y=0$ indicates samples from the one-step model $q_\theta(x_t)$ and $y=1$ indicates data samples from $p_d(x_t)$. With Bayes' rule, we can transform the density ratio estimation problem into a binary classification problem:

$$\frac{q_\theta(x_t)}{p_d(x_t)} \equiv \frac{m(x_t|y=0)}{m(x_t|y=1)} = \frac{p(y=0|x_t)\cancel{m(x_t)}/\cancel{p(y=0)}}{p(y=1|x_t)\cancel{m(x_t)}/\cancel{p(y=1)}} = \frac{p(y=0|x_t)}{p(y=1|x_t)}, \tag{13}$$

where the mixture distribution is defined as $m(x) \equiv m(x_t|y=1)p(y=1) + m(x_t|y=0)p(y=0)$, and the Bernoulli prior distribution $p(y)$ is simply set as a uniform prior $p(y=1) = p(y=0) = 0.5$. In practice, this is achieved by drawing a batch of data from $p_d(x_t)$ with the label $y=1$ and drawing an equal number of samples from $q_\theta(x_t)$ with the label $y=0$. We then train a neural network classifier $c_\eta(x_t,t)$, conditioned on the diffusion time t, to estimate the probability that a given input x_t belongs to class $y=1$. The optimal classifier approximates the posterior probability $c_*(x_t,t) = p(y=1|x_t)$. In this case, the log-density ratio can be written as

$$\nabla_{x_t} \log \frac{q_\theta(x_t)}{p_d(x_t)} = \nabla_{x_t} \log \frac{1-c_*(x_t,t)}{c_*(x_t,t)} = \nabla_{x_t} \mathrm{logit}(1-c_*(x_t,t)). \tag{14}$$

We can then obtain a new DiKL gradient estimator by plugging in the density ratio estimator to Eq. 8:

$$\nabla_\theta \mathrm{DiKL}(q_\theta(x_0)||p_d(x_0)) \approx \sum_{t=1}^{T} w(t) \int q_\theta(x_t) \nabla_{x_t} \mathrm{logit}(1-c_\eta(x_t,t))\frac{\partial x_t}{\partial \theta} dx_t. \tag{15}$$

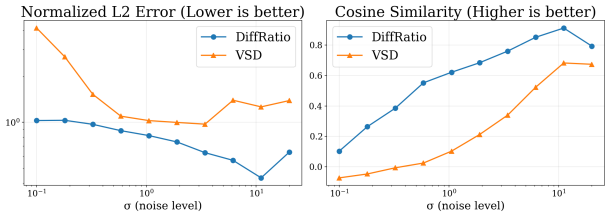


Figure 1: Score difference estimation accuracy on a 2D mixture of Gaussians problem.

Table 1: Evaluation of trained generators on the 2D toy problem. Log-MMD (↓): lower is better. Log-density (↑): higher is better.

METHOD	Log-MMD	Log-density
True	/	-6.65
VSD	-5.25	-10.88
DiffRatio	$\mathbf{-6.08}$	$\mathbf{-6.79}$

Importantly, our method directly estimates the score difference $\nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)$ using a single density ratio estimator $c_\eta(x_t, t)$, which alleviates the two sources of bias in VSD (Eq. 11): (1) Instead of separately estimating the student score $s^{q_\theta}_{\psi_2}(x_t, t)$ and the teacher score $s^{p_d}_{\psi_1}(x_t, t)$, the density ratio estimator directly estimates the score difference in an integrated manner, avoiding the accumulation of errors from two independently trained networks. (2) Without relying on a teacher score for supervision, we avoid bias propagation from the teacher to the student, leading to a *consistent* estimator—given sufficient capacity and training, the estimated score difference converges to the true score difference. In contrast, *VSD cannot guarantee such consistency*, as the bias from teacher supervision cannot be removed regardless of how well the student score network is trained. In the next section, we give an empirical comparison of the gradient estimation bias between VSD and DiffRatio methods.

3.2 Empirical Analysis of the Gradient Bias

To empirically validate that our density-ratio-based gradient estimator reduces bias compared to VSD, we design a controlled 2D experiment where ground-truth score differences can be computed. The data distribution $p_d(x_0)$ is a 2D mixture of Gaussians with 40 components, which allows closed-form computation of the true data score $\nabla_{x_t} \log p_d(x_t)$ at any noise level. The one-step model $q_\theta(x_0)$ is an implicit generative model that has been pre-trained to approximate p_d. Since $q_\theta(x_t)$ lacks a closed-form density, we estimate its score using kernel density estimation (KDE) with 10,000 samples, which provides highly accurate score estimates in this low-dimensional setting. This gives us the ground-truth score difference: $\Delta s^*_t \equiv \Delta s^*(x_t, t) = \nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)$. We compare the following two methods on this problem: (1) DiffRatio (ours) trains a single classifier $c_\eta(x_t, t)$ and estimates the score difference as $\widehat{\Delta s_t} = \nabla_{x_t} \text{logit}(1 - c_\eta(x_t, t))$; (2) VSD trains two separate score networks $s^{p_d}_{\psi_1}$ and $s^{q_\theta}_{\psi_2}$ using DSM and computes their difference $\widehat{\Delta s_t} = s^{q_\theta}_{\psi_2}(x_t, t) - s^{p_d}_{\psi_1}(x_t, t)$.

We evaluate the performance using two metrics: (1) the relative L2 error $\|\widehat{\Delta s_t} - \Delta s^*_t\|_2 / \|\Delta s^*_t\|_2$, and (2) cosine similarity, where $\widehat{\Delta s_t}$ denotes the estimated score difference from each method. We train both methods with 5,000 steps; other details can be found in Appendix D. As shown in Figure 1, our density-ratio-based method consistently achieves lower L2 error and higher cosine similarity than VSD across all noise levels, indicating more accurate gradient estimation. This is because VSD accumulates errors from both the student score network $s^{q_\theta}_{\psi_2}$ and the teacher score network $s^{p_d}_{\psi_1}$. In contrast, our method directly estimates the score *difference* using a single density ratio estimator c_η, avoiding error accumulation from separately trained networks. This empirically validates that density ratio estimation provides more accurate gradient signals for training one-step models. To further illustrate the effect of gradient bias to generation quality, we compare the two gradient estimation methods by training one-step models under the same divergence (DiKL) on this problem. We report log-MMD (maximum mean discrepancy) and log-density under the true data distribution in Table 1. Our method achieves significantly better log-density than VSD, indicating that the generated samples are closer to the true data distribution. The improved gradient estimation translates directly to better generation quality.

3.3 Training Criterion Extensions

In addition to the DiKL, we can use the learned classifier function c_η to obtain a family of gradient estimators for alternative training objectives. For instance, replacing the logit function with the

Algorithm 1 Training one-step diffusion models via DiffRatio (ours)

Require: Training samples $\mathcal{D} \sim p_d(x_0)$

// Stage 1: Pre-train a score model for initialization

Pre-train a score network $s_\theta^{p_d}(x_t, t)$ using DSM (Eq. 9) until convergence

// Stage 2: Train the one-step model without teacher supervision

Create a one-step generator $g_\theta(\cdot) \equiv s_\theta^{p_d}(\cdot, t = t_{\text{init}})$ by fixing the diffusion time argument $t = t_{\text{init}}$ in the pretrained score network

for each training iteration **do**

  Sample time t' for diffusive density ratio estimation

  Train a classifier $c_\eta(x_{t'}, t')$ on samples from $q_\theta(x_{t'}), p_d(x_{t'})$ to estimate their density ratio

  Update the one-step generator g_θ with $c_\eta(x_{t'}, t')$ using Eq. 15 or Eq. 16 or Eq. 17

end for

logarithm function yields an objective that minimizes the probability of generated samples being classified as fake. This formulation aligns with GAN [16, 42] across different diffusion time steps, which approximately minimizes the *Diffusive Jensen-Shannon (DiJS)* divergence (a detailed derivation of DiJS and its gradient estimator can be found in Appendix B):

$$\nabla_\theta \text{DiJS}(q_\theta(x_0)||p_d(x_0)) \approx \sum_{t=1}^{T} w(t) \int q_\theta(x_t) \nabla_{x_t} \log(1 - c_\eta(x_t, t)) \frac{\partial x_t}{\partial \theta} dx_t. \tag{16}$$

Interestingly, the DiJS objective mirrors the minimax GAN formulation, which minimizes the probability of the generated images being fake. Alternatively, one can also maximize the probability of them being real. This approach is referred to as *Diffusive Realism Maximization (DiRM)*, which has the following gradient estimator:

$$\nabla_\theta \text{DiRM}(\theta) \approx -\sum_{t=1}^{T} w(t) \int q_\theta(x_t) \nabla_{x_t} \log c_\eta(x_t, t) \frac{\partial x_t}{\partial \theta} dx_t. \tag{17}$$

Notably, the DiRM objective, which maximizes the likelihood of the generated images being real, mirrors the *non-saturating GAN* formulation [16], which is known to provide more stable gradients for the generator compared to the original minimax objective.

In principle, once the density ratio is available, any ratio-based divergence (e.g., f-divergence [64]) can be used for training. This flexibility is not available when one only has access to the teacher and student scores (as in VSD), making DiffRatio applicable to a broader family of criteria. Moreover, we provide a convergence discussion in Appendix C, where we justify the consistency of our DiffRatio gradient estimators under *weaker assumptions* than those required for standard ratio estimation [24].

3.4 Simplified Training Procedure

The overall training procedure of our proposed DiffRatio framework is summarized in Algorithm 1. In our method, the generator is first initialized with a pre-trained score network and then further trained as a one-step model with the diffusion time argument in the score network fixed to $t{=}t_{\text{init}}$. This design leads to a more simplified training pipeline compared to VSD. Specifically, VSD requires maintaining and evaluating three networks during training: (1) the pre-trained teacher score network $s_{\psi_1}^{p_d}$, (2) the student score network $s_{\psi_2}^{q_\theta}$, and (3) the generator g_θ. In contrast, our method only requires two: (1) the generator g_θ, and (2) the density ratio estimator c_η (which is approximately half the size of a full score network). This significantly reduces the size of the auxiliary networks during training.

For low-dimensional problems such as the 2D mixture of Gaussians in Section 3.2, the generator can be trained from random initialization. However, for high-dimensional image generation, we find that initializing the generator with a pre-trained score network is necessary to prevent mode collapse, which is consistent with findings in the score-based distillation literature [36, 72, 73]. To further understand the necessity of this pre-training, we provide a detailed analysis in Section 6, which studies the role of weight initialization from two perspectives: (1) the functional mapping perspective, and (2) the feature space perspective. Our analysis shows that multi-level features learned during diffusion pre-training are essential for preventing mode collapse in high-dimensional settings.

3.5 Inference-Time Parallel Scaling with Learned Density Ratio Network

Unlike existing distillation approaches [36, 72, 73] where the learned student score $s_{\psi_1}^{q_\theta}(x_t, t)$ network is rarely useful after training, the learned density ratio network in our approach can be leveraged at inference time as a judge to further refine generation quality. Specifically, to make the samples from the trained model $q_\theta(x)$ closer to samples from $p_d(x)$, we noticed that

$$p_d(x_0) = q_\theta(x_0)\frac{p_d(x_0)}{q_\theta(x_0)} = q_\theta(x_0)\frac{c_*(x_0)}{1 - c_*(x_0)}, \tag{18}$$

which allows us to perform *importance resampling* to correct the samples with our consistent ratio estimator. Similar constructions have been explored in the GAN–EBM literature [9, 1].

In practice, we first generate M candidates from our one-step model: $x_0^{(1:M)} \sim q_\theta(x_0)$, then we select one sample $x_0^{(i)}$ from $\{x_0^{(1:M)}\}$ according to the categorical distribution

$$i \sim \mathrm{Cat}\left(\frac{w(x_0^{(1)})}{\sum_{j=1}^M w(x_0^{(j)})}, \cdots, \frac{w(x_0^{(M)})}{\sum_{j=1}^M w(x_0^{(j)})}\right), \tag{19}$$

where the density ratio w is directly approximated by the learned ratio at t_{min} to avoid the indefiniteness of the density ratio for manifold distributions [56]:

$$w(x_0^{(i)}) \equiv \frac{p_d(x_0^{(i)})}{q_\theta(x_0^{(i)})} \approx \frac{c_\eta(x_{t_{\text{min}}}^{(i)}, t_{\text{min}})}{1 - c_\eta(x_{t_{\text{min}}}^{(i)}, t_{\text{min}})}. \tag{20}$$

This procedure enables effective inference-time scaling: increasing M monotonically reduces the sampling bias of the importance-resampled estimator of the target distribution. Crucially, all M samples and their corresponding ratio evaluations can be computed fully *in parallel*, making this inference-time scaling step highly efficient in practice.

4 Related Work

Density ratio estimation lies at the core of many GAN variants [16, 42]. In classic GANs, the discriminator implicitly estimates the density ratio between the data and model distributions. However, in high-dimensional image modeling, both distributions typically concentrate on low-dimensional manifolds and are not absolutely continuous, making the density ratio—and consequently the Jensen–Shannon divergence—ill-defined. This issue is widely recognized as a major source of GAN training instability [2, 3, 37, 48]. To address this, Wasserstein GANs replace the JS divergence with the Wasserstein-1 distance, which remains well-defined even when the distributions are disjoint [3]. In practice, enforcing the required 1-Lipschitz constraint is challenging and relies on heuristic regularization methods such as weight clipping [3], gradient penalties [17], or spectral normalization [39]. As a result, the optimized objective often deviates from the ideal divergence, and empirical stability is largely driven by regularization rather than strict divergence minimization [37, 12].

Adding Gaussian noise to real and fake samples has been proposed as a way to ensure distributions are fully supported, making the density ratio well-defined [52, 48, 42, 68]. This model-agnostic approach requires no architectural changes but hinges on choosing an effective noise level—something hard to fix throughout training. Diffusion GAN [60] addresses the challenge of selecting a fixed noise level by introducing a diffusion-inspired noise schedule that gradually increases noise in tandem with the model's learning capacity. While this represents the most closely related work to ours, our approach differs in some important aspects.

First, DiffRatio is agnostic to the generator's network architecture. Diffusion GAN stabilizes training using a StyleGAN-based generator [26], which is implicitly trained progressively from low to high resolutions while keeping the network topology fixed. In contrast, DiffRatio does not rely on a specialized generator architecture or progressive resolution training. Instead, we adopt a generic U-Net architecture, resulting in a simpler and more broadly applicable framework. Second, DiffRatio does not rely on additional regularization techniques or GAN-specific training tricks such as gradient penalties or spectral normalization. Therefore, we provide a clean framework for training one-step diffusion models yet still achieves stable convergence and demonstrates training robustness.

Table 2: Unconditional one-step generation on CIFAR-10.

METHOD	NFE (↓)	FID (↓)	IS (↑)
Teacher model			
EDM [25]	35	2.04	9.84
EDM [25]	1	8.70	8.49
Training from scratch without teacher			
CT [55]	1	8.70	8.49
iCT [54]	1	2.83	9.54
iCT-deep [54]	1	**2.51**	**9.76**
BCM [30]	1	3.10	9.45
BCM-deep [30]	1	2.64	9.67
sCT [34]	1	2.85	-
Diffusion-GAN [60]	1	3.19	-
IMM [71]	1	3.20	-
MeanFlow [14]	1	2.92	-
Distillation with teacher supervision			
Progressive Distillation [49]	1	8.34	8.69
DSNO [70]	1	3.78	-
TRACT [6]	1	3.78	-
CD [55]	1	3.55	9.48
CTM (w/ GAN) [27]	1	1.98	-
CTM (w/o GAN) [27]	1	>5.0	-
sCD [34]	1	3.66	-
Diff-Instruct [36]	1	4.53	-
ECT [15]	1	3.60	-
SiD [73]	1	2.03	10.02
SiDA [72]	1	**1.52**	**10.32**
Training without teacher supervision (ours)			
DiffRatio-DiRM	1	4.87	9.85
DiffRatio-DiKL	1	3.81	9.90
DiffRatio-DiJS	1	**2.39**	**9.93**

Table 3: Conditional one-step generation on ImageNet 64×64.

METHOD	NFE (↓)	FID (↓)
Teacher model		
EDM [25]	79	2.44
Training from scratch without teacher		
EDM2-L/XL [26]	1	13.0
CT [55]	1	13.0
iCT [54]	1	4.02
iCT-deep [54]	1	3.25
BCM [30]	1	4.18
BCM-deep [30]	1	3.14
sCT [34]	1	**2.04**
Distillation with teacher supervision		
Progressive Distillation [49]	1	7.88
DSNO [70]	1	7.83
TRACT [6]	1	7.43
CD [55]	1	6.20
CTM (w/ GAN) [27]	1	1.92
sCD [34]	1	2.44
Diff-Instruct [36]	1	5.57
EM Distillation [63]	1	2.20
ECT [15]	1	2.49
SiD [73]	1	2.02
SiDA [72]	1	**1.35**
DMD2 (w/ GAN) [65]	1	1.51
Training without teacher supervision (ours)		
DiffRatio-DiJS	1	**1.54**

Therefore, our method leads to a conceptually cleaner training objective that is more amenable to theoretical analysis and empirical diagnosis, as it avoids the complexities and instabilities associated with specific network architectures or additional regularization tricks.

5 Experiments

We evaluate the performance of our proposed method by training one-step image diffusion models on two standard benchmarks: (1) unconditional generation on the CIFAR-10 (32×32) dataset [29], (2) class-label-conditioned generation on the ImageNet (64×64 and 512×512) dataset [10]. Our one-step diffusion model adopts the same neural network architecture as the pre-trained score model. Specifically, we employ the EDM U-Net model architecture [25] for CIFAR-10 and ImageNet 64×64, and use the EDM2-M U-Net model architecture [26] for ImageNet 512×512. We consider three training criteria in our framework (DiKL/DiJS/DiRM) on CIFAR-10, and use the best-performing criterion (DiJS) for higher-dimensional ImageNet experiments (64×64 and 512×512). We use the variance-exploding (VE) noise schedule to define the diffusive divergences. Our density ratio estimator $c_\eta(x_t, t)$ is implemented using the encoder portion of the teacher's U-Net architecture with a sigmoid function attached to the end to produce a scalar output between 0 and 1. This network is approximately half the size of the full U-Net used for score estimation, leading to reduced size of auxiliary networks. The one-step generator $g_\theta(z)$ is initialized using the weights from the pre-trained score model with the diffusion time argument $t_{\text{init}}=2.5$ fixed throughout training and sampling. We follow standard hyperparameter settings for training generative models on CIFAR-10 and ImageNet as detailed in [25, 26]. We use a batch size of 64 for CIFAR-10 and 4,096 for ImageNet. The weighting function $w(t)=\sigma_t^2$ is used for for both datasets. We adopt a single-step update strategy for the density ratio estimator throughout our experiments, consistent with previous works [36, 73]. See Appendix E for more details.

In Tables 2, 3 and 4, we compare our method to previous methods for training one-step generative models. To highlight methodological differences, we categorize these approaches into three groups: (1) training from scratch, (2) distillation with teacher supervision (e.g., using teacher score, denoiser or ODE trajectory as supervision in the distillation loss), and (3) training without teacher supervision.



Figure 2: Images generated by DiffRatio-DiJS-M on ImageNet 512×512 (FID=1.41) with 1 NFE.

Table 4: Conditional one-step generation on ImageNet 512×512. † indicates that NFE is executed *in parallel* rather than sequentially.

METHOD	CFG	NFE (↓)	FID (↓)
Teacher model			
EDM2-M [26]	N	63	2.25
EDM2-M [26]	Y	126	2.01
Training from scratch without teacher			
sCT-M [34]	Y	1	5.84
sCT-M [34]	Y	2	5.53
sCD-M [34]	Y	1	**2.75**
sCD-M [34]	Y	2	**2.26**
Distillation with teacher supervision			
SiD-M [73]	N	1	2.06
SiDA-M [72]	N	1	**1.55**
Training without teacher supervision (ours)			
DiffRatio-DiJS-M	N	1	**1.41**
DiffRatio-DiJS-M (w/ inference-time scaling)	N	$10^{\dagger}$	1.35
DiffRatio-DiJS-M (w/ inference-time scaling)	N	$20^{\dagger}$	1.34
DiffRatio-DiJS-M (w/ inference-time scaling)	N	$50^{\dagger}$	1.34
DiffRatio-DiJS-M (w/ inference-time scaling)	N	$100^{\dagger}$	**1.32**

We find that our proposed method, DiffRatio, achieves competitive one-step generation performance despite not using any teacher supervision, comparable to most state-of-the-art distillation methods that rely on full teacher supervision. The only baseline method with similar performance to ours is SiDA [72], which depends on training data, teacher score supervision, teacher weight initialization, and student score estimation. In contrast, our method requires only training data, teacher weight initialization, and density ratio estimation. Notably, this *eliminates* the need for teacher supervision in the training process. Also, density ratio estimation is both simpler and *more lightweight* than student score estimation, as the density ratio network is approximately half the size of a full score network. Moreover, unlike VSD, our approach does not require maintaining or evaluating the teacher score network during the training process. This results in a more streamlined training framework, since our approach does not require maintaining and evaluating the teacher and student score networks in the training procedure. Images generated by DiffRatio-DiJS are visualized in Figures 2. More generated images can be found in Appendix F (Figures 6, 7 and 8).

Inference-Time Parallel Scaling. We further explore the inference-time refinement of our approach using the learned ratio, as we discussed in Section 3.5. We first investigate the following question: *after training, is there still a discrepancy between the distributions of the real and generated data?* We evaluate the learned density ratio on both real and generated images by Eq. 20 and visualize the resulting distributions via histograms in Figure 3. We can see that the two distributions largely overlap, indicating that our generator achieves a relatively good performance. However, real images still exhibit a slightly larger ratio on average than generated ones, suggesting a small but consistent mismatch between $q_\theta(x_0)$ and $p_d(x_0)$, captured by the learned ratio. We therefore investigate inference-time scaling via importance resampling using the learned ratio on ImageNet 512×512. In Table 4, we report results with candidate set sizes $M \in \{10, 20, 50, 100\}$. The FID decreases monotonically as we allocate more inference-time compute, resulting in a significant improvement from 1.41 to 1.32.

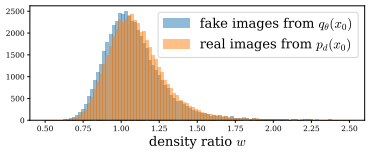
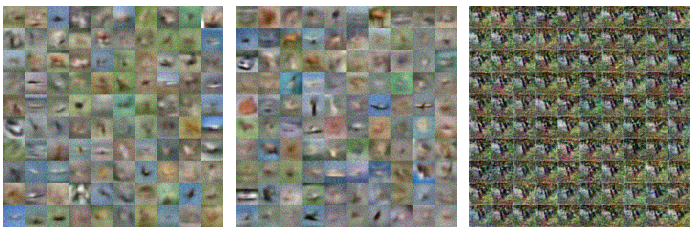


Figure 3: Histograms of the learned density ratio w.

6 Analysis of the Role of Weight Initialization in One-Step Diffusion Models

Our one-step diffusion model was initialized with the pre-trained score model's weights. We observed that training from random initialization led to mode collapse; see Figure 4c for an example of mode collapse. One possible explanation is that mode collapse arises from the training objectives (i.e.,



(a) Single-level Init. (b) Multi-level Init. (c) Collapsed Samples

Figure 5: Visualization of samples from models trained with different initializations and collapsed samples on CIFAR-10.

Table 5: One-step models trained by DiJS with different initializations on various CIFAR subsets.

Initialization	Initialization dataset	FID
No init.	–	collapsed
Single-level DSM init.	full CIFAR-10	collapsed
Multi-level DSM init.	10 classes in CIFAR-100	collapsed
	50 classes in CIFAR-100	6.20
	90 classes in CIFAR-100	6.01
	full CIFAR-10	2.39

reverse KL or JS divergence), a phenomenon also observed in GAN literature [16]. To understand why initializing the one-step model with the pre-trained score model's weights prevents mode collapse in the training process, we investigate the following two hypotheses.

Function Space Hypothesis. *Teacher weight initialization provides a more structured latent-to-output functional mapping, i.e., different locations in the latent space are initially mapped to distinct images, preventing mode collapse.*

This hypothesis originally arose from visualizing initialized samples as shown in Figure 4a, showing that initialization already induces diverse mappings, with the one-step model training stage primarily refining these initializations into sharper images. Somewhat surprisingly, however, we find that functional initialization alone is insufficient to prevent mode collapse. To show this, instead of training the teacher model across different diffusion time steps t and selecting a single time step t_{init} for initialization, we only pre-train the teacher model at the selected time step t_{init} and use its weight to initialize the one-step model. This setup ensures identical latent-to-output mappings for the one-step model at initialization as shown in Figure 4b. However, with this initialization, the one-step model still exhibits mode collapse early in the one-step model training stage, which suggests that the functional mapping perspective alone does not fully explain the mode-collapse issue.

Feature Space Hypothesis. *Teacher weight initialization provides a rich set of multi-level features learned when pre-training the teacher diffusion model, which help prevent mode collapse.*

To verify this hypothesis and isolate the role of learned features from functional mapping effects, we pre-train the teacher model on CIFAR-100 while excluding all classes that overlap with CIFAR-10. This ensures that images from the target classes that the one-step model aims to generate are absent during pre-training, allowing us to focus solely on the contribution of the learned features. We train the teacher model using increasingly larger subsets of CIFAR-100 with (10, 50, 90) classes, creating a setting with increasing feature diversity. Table 5 shows the performance of our one-step model on CIFAR-10 initialized with the weights of teacher models trained on varying numbers of CIFAR-100 classes. We find that when the teacher model is trained on only 10 classes, mode collapse still occurs. However, as the number of training classes increases, the one-step model no longer collapses, indicating that feature richness plays a crucial role in preventing mode collapse. Nevertheless, despite mitigating mode collapse, this initialization strategy achieves an FID of 6.01 when the teacher model is pre-trained on all 90 non-overlapping classes in CIFAR-100, which is significantly worse than the FID (2.39) obtained when directly using CIFAR-10 as the pre-training dataset. This suggests that while feature richness is essential for stabilizing training, functional mapping initialization remains important for achieving higher sample quality.

7 Conclusions

This paper introduced **DIFFRATIO**, a new approach for training one-step diffusion models by directly estimating the score difference between the one-step model and the true data distribution across diffusion time steps with a lightweight density ratio network. Our approach mitigates the gradient estimation bias in score-based distillation and reduces the size of auxiliary networks. Furthermore, the learned density ratio network can be leveraged to unlock a principled inference-time parallel scaling without external rewards or additional sequential computation. DiffRatio achieved competitive generation quality on image synthesis benchmarks, outperforming most approaches with full teacher supervision. Limitations and future work are discussed in Appendix H.

Acknowledgments and Disclosure of Funding

We would like to thank Prof. Mingyuan Zhou for the useful discussion. MZ and DB acknowledge funding from AI Hub in Generative Models, under grant EP/Y028805/1. DB is also supported by the Engineering and Physical Sciences Research Council (grant number UKRI5028). JH is supported by the University of Cambridge Harding Distinguished Postgraduate Scholars Programme. ZO is supported by the Lee Family Scholarship. JMHL acknowledges support from a Turing AI Fellowship under grant EP/V023756/1. Part of the work was done when WC was a PhD student at University of Cambridge and Max Planck Institute for Intelligent Systems.

References

[1] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv preprint arXiv:2003.05033*, 2020.

[2] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[4] Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. *arXiv preprint arXiv:2206.07309*, 2022.

[5] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*, 2022.

[6] David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbott, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

[7] Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025.

[8] Valentin De Bortoli, Alexandre Galashov, J. Swaroop Guntupalli, Guangyao Zhou, Kevin Murphy, Arthur Gretton, and Arnaud Doucet. Distributional diffusion models with scoring rules, 2025.

[9] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287, 2020.

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

[11] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[12] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. *arXiv preprint arXiv:1710.08446*, 2017.

[13] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

[14] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[15] Zhengyang Geng, Ashwini Pokle, Weijian Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[18] 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, 2010.

[19] Jiajun He, Wenlin Chen, Mingtian Zhang, David Barber, and José Miguel Hernández-Lobato. Training neural samplers with reverse diffusive kl divergence. *arXiv preprint arXiv:2410.12456*, 2024.

[20] Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.

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

[22] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[23] Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

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

[25] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

[26] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024.

[27] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] Yeongmin Kim, Byeonghu Na, Minsang Park, JoonHo Jang, Dongjun Kim, Wanmo Kang, and Il-Chul Moon. Training unbiased diffusion models from biased dataset. In *The Twelfth International Conference on Learning Representations*, 2024.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

[30] Liangchen Li and Jiajun He. Bidirectional consistency models. *arXiv preprint arXiv:2403.18035*, 2024.

[31] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

[32] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

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

[34] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *International Conference on Learning Representations*, 2025.

[35] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[36] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[38] Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

[39] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[40] Byeonghu Na, Yeongmin Kim, Minsang Park, Donghyeok Shin, Wanmo Kang, and Il-Chul Moon. Diffusion rejection sampling. In *International Conference on Machine Learning*, pages 37097–37121. PMLR, 2024.

[41] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[42] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

[43] Zijing Ou, Mingtian Zhang, Andi Zhang, Tim Z Xiao, Yingzhen Li, and David Barber. Improving probabilistic diffusion models with optimal covariance matching. *International Conference on Learning Representations*, 2025.

[44] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2022.

[45] Jing Qin. Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630, 1998.

[46] Herbert E Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394. Springer, 1992.

[47] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[48] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. *Advances in neural information processing systems*, 30, 2017.

[49] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[50] Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching. *arXiv preprint arXiv:2406.04103*, 2024.

[51] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[52] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

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

[54] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

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

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

[57] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

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

[59] Leyang Wang, Mingtian Zhang, Zijing Ou, and David Barber. Vardiu: A variational diffusive upper bound for one-step diffusion distillation. *arXiv preprint arXiv:2508.20646*, 2025.

[60] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-GAN: Training GANs with diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[61] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[62] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

[63] Sirui Xie, Zhisheng Xiao, Diederik P Kingma, Tingbo Hou, Ying Nian Wu, Kevin Patrick Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. Em distillation for one-step diffusion models. *arXiv preprint arXiv:2405.16852*, 2024.

[64] Yilun Xu, Weili Nie, and Arash Vahdat. One-step diffusion models with f-divergence distribution matching. *arXiv preprint arXiv:2502.15681*, 2025.

[65] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024.

[66] Longlin Yu, Tianyu Xie, Yu Zhu, Tong Yang, Xiangyu Zhang, and Cheng Zhang. Hierarchical semi-implicit variational inference with application to diffusion model acceleration. *Advances in Neural Information Processing Systems*, 36, 2024.

[67] Mingtian Zhang, Thomas Bird, Raza Habib, Tianlin Xu, and David Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019.

[68] Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib, and David Barber. Spread divergence. In *International Conference on Machine Learning*, pages 11106–11116. PMLR, 2020.

[69] Mingtian Zhang, Andi Zhang, Tim Z Xiao, Yitong Sun, and Steven McDonagh. Out-of-distribution detection with class ratio estimation. *arXiv preprint arXiv:2206.03955*, 2022.

[70] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *International conference on machine learning*, pages 42390–42402. PMLR, 2023.

[71] Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025.

[72] Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. In *The Thirteenth International Conference on Learning Representations*, 2025.

[73] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

A Diffusive Kullback–Leibler (DiKL) Divergence

A.1 Validity of DiKL Divergence

We follow the derivation from Spread Divergence [68] and provide a simple proof of the validity of the diffusive KL (DiKL) divergence. See [68] for a generalized proof that includes cases with non-absolutely continuous distributions and non-Gaussian kernels.

Recall that DiKL is defined as

$$\mathrm{DiKL}_{\mathcal{K}}(q_\theta(x_0)||p_d(x_0)) = \sum_{t=1}^{T} w(t)\,\mathrm{KL}(q_\theta(x_t)||p_d(x_t)), \tag{21}$$

where $w(t) > 0$, and the noisy model density $q_\theta(x_t)$ and noisy data density $p_d(x_t)$ are respectively defined as:

$$q_\theta(x_t) = \int k_t(x_t|x_0) q_\theta(x_0) dx_0, \tag{22}$$

$$p_d(x_t) = \int k_t(x_t|x_0) p_d(x_0) dx_0, \tag{23}$$

with $k_t(x_t|x_0)$ denoting the Gaussian transition kernel. We will show that

$$\mathrm{DiKL}_{\mathcal{K}}(q_\theta(x_0)||p_d(x_0)) = 0 \iff q_\theta(x_0) = p_d(x_0). \tag{24}$$

Since $w(t) > 0$ and the KL divergence is non-negative, it suffices to show that:

$$\mathrm{KL}(q_\theta(x_t)||p_d(x_t)) = 0 \iff q_\theta(x_0) = p_d(x_0). \tag{25}$$

First, by definition of KL divergence, we have

$$\mathrm{KL}(q_\theta(x_t)||p_d(x_t)) = 0 \iff q_\theta(x_t) = p_d(x_t). \tag{26}$$

Now, assume that $k_t(\epsilon) = \mathcal{N}(0, \sigma^2 I)$, and rewrite the noisy densities as convolutions:

$$q_\theta(x_t) = (q_\theta * k_t)(x_t), \tag{27}$$

$$p_d(x_t) = (p_d * k_t)(x_t). \tag{28}$$

Suppose $q_\theta(x_t) = p_d(x_t)$. Applying the Fourier transform $\mathcal{F}$, we obtain:

$$\mathcal{F}(q_\theta * k_t) = \mathcal{F}(q_\theta) \cdot \mathcal{F}(k_t), \tag{29}$$

$$\mathcal{F}(p_d * k_t) = \mathcal{F}(p_d) \cdot \mathcal{F}(k_t). \tag{30}$$

Since the characteristic function of Gaussian $\mathcal{F}(k_t) > 0$, we have:

$$q_\theta(x_t) = p_d(x_t) \iff \mathcal{F}(q_\theta) \cdot \cancel{\mathcal{F}(k_t)} = \mathcal{F}(p_d) \cdot \cancel{\mathcal{F}(k_t)} \tag{31}$$

$$\iff \mathcal{F}(q_\theta) = \mathcal{F}(p_d) \tag{32}$$

$$\iff q_\theta(x_0) = p_d(x_0). \tag{33}$$

This competes the proof.

A.2 Derivation of the Density-Ratio Gradient Estimator for DiKL Divergence

Without loss of generality, we first consider DiKL with a single kernel k_t. We will first follow [19] and show that the gradient of DiKL w.r.t. the model parameter θ is given by

$$\nabla_\theta \mathrm{DiKL}_{k_t}(q_\theta||p_d) = \int q_\theta(xt) \left(\nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)\right) \frac{\partial x_t}{\partial \theta} dx_t. \tag{34}$$

Recall that the DiKL divergence for a single kernel k_t is defined as

$$\mathrm{DiKL}_{k_t}(q_\theta||p_d) = \int \left(\log q_\theta(x_t) - \log p_d(x_t)\right) q_\theta(x_t) dx_t. \tag{35}$$

We first reparameterize x_t as a function of z and ϵ:

$$x_t = \alpha_t g_\theta(z) + \sigma_t \epsilon_t \equiv h_\theta(z, \epsilon_t), \tag{36}$$

where $z \sim p(z) \equiv \mathcal{N}(z|0, I)$ and $\epsilon_t \sim p(\epsilon_t) \equiv \mathcal{N}(\epsilon_t|0, I)$. It then follows that

$$\nabla_\theta \mathrm{DiKL}_{k_t}(q_\theta(x_0)||p_d(x_0)) = \nabla_\theta \int (\log q_\theta(x_t) - \log p_d(x_t))\, q_\theta(x_t) dx_t \tag{37}$$

$$= \nabla_\theta \iiint (\log q_\theta(x_t) - \log p_d(x_t))\, \delta(x_t - h_\theta(z, \epsilon_t)) p(z) p(\epsilon_t) dx_t dz d\epsilon \tag{38}$$

$$= \nabla_\theta \iint (\log q_\theta(x_t) - \log p_d(x_t))\, |_{x_t = h_\theta(z,\epsilon_t)} p(z) p(\epsilon_t) dz d\epsilon \tag{39}$$

$$= \int \left(\nabla_\theta \log q_\theta(x_t) + \nabla_{x_t} \log q_\theta(x_t) \frac{\partial x_t}{\partial \theta} - \nabla_{x_t} \log p_d(x_t) \frac{\partial x_t}{\partial \theta} \right) q_\theta(x_t) dx_t \tag{40}$$

$$= \int \left(\nabla_{x_t} \log q_\theta(x_t) \frac{\partial x_t}{\partial \theta} - \nabla_{x_t} \log p_d(x_t) \frac{\partial x_t}{\partial \theta} \right) q_\theta(x_t) dx_t, \tag{41}$$

where the last line follows since

$$\int \nabla_\theta \log q_\theta(x_t) q_\theta(x_t) dx_t = \int \nabla_\theta\, q_\theta(x_t) dx_t = \nabla_\theta \int q_\theta(x_t) dx_t = \nabla_\theta 1 = 0. \tag{42}$$

For a set of kernels $\mathcal{K} = \{k_1, \cdots, k_T\}$, it then follows that

$$\nabla_\theta \mathrm{DiKL}_{\mathcal{K}}(q_\theta||p_d) = \sum_{t=1}^{T} w(t) \int q_\theta(x_t) \left(\nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t) \right) \frac{\partial x_t}{\partial \theta} dx_t \tag{43}$$

$$= \sum_{t=1}^{T} w(t) \int q_\theta(x_t) \left(\nabla_{x_t} \log \frac{q_\theta(x_t)}{p_d(x_t)} \right) \frac{\partial x_t}{\partial \theta} dx_t \tag{44}$$

$$= \sum_{t=1}^{T} w(t) \int q_\theta(x_t) \nabla_{x_t} \,\mathrm{logit}(1 - c_*(x_t, t)) \frac{\partial x_t}{\partial \theta} dx_t, \tag{45}$$

where the last line follows by Eq. 14. This completes the proof.

B Diffusive Jensen-Shannon (DiJS) Divergence

B.1 Validity of DiJS Divergence

We follow a similar derivation for the validity of DiKL in Appendix A.1. Recall that DiJS is defined as

$$\mathrm{DiJS}_{\mathcal{K}}(q_\theta(x_0)||p_d(x_0)) = \sum_{t=1}^{T} w(t)\, \mathrm{JS}(q_\theta(x_t)||p_d(x_t)), \tag{46}$$

where $w(t) > 0$, and the noisy model density $q_\theta(x_t)$ and noisy data density $p_d(x_t)$ are respectively defined as:

$$q_\theta(x_t) = \int k_t(x_t|x_0) q_\theta(x_0) dx_0, \tag{47}$$

$$p_d(x_t) = \int k_t(x_t|x_0) p_d(x_0) dx_0, \tag{48}$$

with $k_t(x_t|x_0)$ denoting the Gaussian transition kernel. We will show that

$$\mathrm{DiJS}_{\mathcal{K}}(q_\theta(x_0)||p_d(x_0)) = 0 \iff q_\theta(x_0) = p_d(x_0). \tag{49}$$

Since $w(t) > 0$ and the JS divergence is non-negative, it suffices to show that:

$$\mathrm{JS}(q_\theta(x_t)||p_d(x_t)) = 0 \iff q_\theta(x_0) = p_d(x_0). \tag{50}$$

First, by definition of JS divergence, we have

$$\mathrm{JS}(q_\theta(x_t)||p_d(x_t)) = 0 \iff q_\theta(x_t) = p_d(x_t). \tag{51}$$

Now, assume that $k_t(\epsilon) = \mathcal{N}(0, \sigma^2 I)$, and rewrite the noisy densities as convolutions:

$$q_\theta(x_t) = (q_\theta * k_t)(x_t), \tag{52}$$

$$p_d(x_t) = (p_d * k_t)(x_t). \tag{53}$$

Suppose $q_\theta(x_t) = p_d(x_t)$. Applying the Fourier transform $\mathcal{F}$, we obtain:

$$\mathcal{F}(q_\theta * k_t) = \mathcal{F}(q_\theta) \cdot \mathcal{F}(k_t), \tag{54}$$

$$\mathcal{F}(p_d * k_t) = \mathcal{F}(p_d) \cdot \mathcal{F}(k_t). \tag{55}$$

Since the characteristic function of Gaussian $\mathcal{F}(k_t) > 0$, we have:

$$q_\theta(x_t) = p_d(x_t) \iff \mathcal{F}(q_\theta) \cdot \cancel{\mathcal{F}(k_t)} = \mathcal{F}(p_d) \cdot \cancel{\mathcal{F}(k_t)} \tag{56}$$

$$\iff \mathcal{F}(q_\theta) = \mathcal{F}(p_d) \tag{57}$$

$$\iff q_\theta(x_0) = p_d(x_0). \tag{58}$$

This competes the proof.

B.2 Derivation of the Density-Ratio Gradient Estimator for DiJS Divergence

Recall that the Jensen–Shannon (JS) divergence between two densities q and p is

$$\mathrm{JS}(q \,\|\, p) = \frac{1}{2}\mathrm{KL}\left(q \,\middle\|\, \frac{q+p}{2}\right) + \frac{1}{2}\mathrm{KL}\left(p \,\middle\|\, \frac{q+p}{2}\right). \tag{59}$$

Define the mixture density at diffusion time t as

$$m(x_t) \equiv \frac{q_\theta(x_t) + p_d(x_t)}{2}. \tag{60}$$

We first consider a single diffusion kernel k_t and then extend the result to the weighted multi-kernel setting.

Density-ratio identity via an optimal classifier. Following the GAN view [16], consider a time-conditional binary classification problem between data and model samples at diffusion time t, where $y = 1 \sim p_d(x_t)$ and $y = 0 \sim q_\theta(x_t)$, let the Bayes-optimal classifier be $c_*(x_t, t) \equiv p(y = 1 \mid x_t, t)$. Then

$$c_*(x_t, t) = \frac{p_d(x_t)}{p_d(x_t) + q_\theta(x_t)}, \qquad 1 - c_*(x_t, t) = \frac{q_\theta(x_t)}{p_d(x_t) + q_\theta(x_t)}. \tag{61}$$

Consequently, the density ratio appearing in $\mathrm{KL}(q_\theta \| m)$ can be written as

$$\frac{q_\theta(x_t)}{m(x_t)} \;=\; \frac{q_\theta(x_t)}{\frac{1}{2}(p_d(x_t) + q_\theta(x_t))} \;=\; 2\frac{q_\theta(x_t)}{p_d(x_t) + q_\theta(x_t)} \;=\; 2\big(1 - c_*(x_t, t)\big). \tag{62}$$

Gradient of the DiJS objective under the GAN approximation. Consider the first JS term:

$$\frac{1}{2}\mathrm{KL}(q_\theta(x_t) \,\|\, m(x_t)) = \frac{1}{2}\int q_\theta(x_t) \log\left(\frac{q_\theta(x_t)}{m(x_t)}\right) dx_t. \tag{63}$$

Under the standard GAN approximation, we treat the density-ratio estimate (equivalently the optimal classifier) as fixed when differentiating w.r.t. θ, i.e., we detach $m(x_t)$.[2] Using the same reparameterization argument as in Appendix A.2, we obtain

$$\nabla_\theta \frac{1}{2}\mathrm{KL}(q_\theta(x_t) \,\|\, m(x_t)) \approx \frac{1}{2}\int q_\theta(x_t)\, \nabla_{x_t} \log\left(\frac{q_\theta(x_t)}{m(x_t)}\right) \frac{\partial x_t}{\partial \theta}\, dx_t. \tag{64}$$

Substituting (62) and dropping the constant $\log 2$ (whose gradient w.r.t. x_t is zero), this becomes

$$\nabla_\theta \frac{1}{2}\mathrm{KL}(q_\theta(x_t) \,\|\, m(x_t)) \approx \frac{1}{2}\int q_\theta(x_t)\, \nabla_{x_t} \log\big(1 - c_*(x_t, t)\big) \frac{\partial x_t}{\partial \theta}\, dx_t. \tag{65}$$

For the second JS term, $\frac{1}{2}\mathrm{KL}(p_d \| m)$, the data density p_d does not depend on θ, and under the same detach-m approximation its contribution to ∇_θ is ignored. Therefore, the resulting gradient estimator is given by (65).

[2] This is justified by the envelope theorem [38] for the inner maximization over the classifier parameters, and corresponds to a two-timescale training regime where the classifier tracks its optimum for the current θ. See also Theorem 2.4 in [2] for a discussion of vanishing gradients in the GAN setting.

C Discussion on the Consistency of the Diffusive Density-Ratio Estimator

Appendices A and B establish that DiKL and DiJS are valid divergences: a zero value of either divergence implies equality of the original model and data distributions. This identifies the desired population optimum. To justify the classifier-based score-difference estimates used in our gradient formulas, however, we must distinguish consistency of the estimated log-density-ratio values from consistency of their input gradients. The latter requires an additional regularity condition, which we state and prove below.

Consistency of the log-density-ratio values. Throughout this discussion, we will fix the generator parameters θ and a noise level $t > 0$. We train a time-conditional logistic classifier $c_\eta(x_t, t)$ to distinguish real samples from generated samples, using equal class priors. The population-optimal classifier is

$$c^\star(x_t, t) = \frac{p_d(x_t)}{p_d(x_t) + q_\theta(x_t)}. \tag{66}$$

Consequently, its log odds recover the log-density ratio:

$$\ell(x_t, t) := \log \frac{q_\theta(x_t)}{p_d(x_t)} = \log \frac{1 - c^\star(x_t, t)}{c^\star(x_t, t)}. \tag{67}$$

Let $\{\hat{\eta}_n\}_{n\geq 1}$ be a sequence of classifier parameter estimates, where n indexes the estimators and may represent classifier training iterations with the generator parameters θ held fixed. Define

$$\widehat{\ell}_n(x_t, t) := \log \frac{1 - c_{\hat{\eta}_n}(x_t, t)}{c_{\hat{\eta}_n}(x_t, t)}. \tag{68}$$

Under suitable statistical and optimization conditions, a well-specified logistic classifier trained by maximum likelihood consistently estimates the class probabilities and hence the log-density ratio; see the related NCE consistency result in Theorem 2 of [18]. However, consistency of the estimated values alone does not imply consistency of their input gradients. The following theorem gives a sufficient additional smoothness condition for gradient consistency.

Theorem C.1 (Gradient consistency of the diffusive density-ratio estimator)**.** *Fix θ and $t > 0$, and let ℓ and $\widehat{\ell}_n$ be the log-density ratio and its estimate defined as above. Suppose that $\ell(\cdot, t)$ is continuously differentiable and that the following conditions hold:*

1. ***Pointwise log-density-ratio consistency.*** *For every x, we have*

$$\widehat{\ell}_n(x, t) \xrightarrow{\mathbb{P}} \ell(x, t) \quad \text{as } n \to \infty. \tag{69}$$

2. ***Uniform local regularity of the estimated gradients.*** *Each $\widehat{\ell}_n(\cdot, t)$ is continuously differentiable almost surely. For every compact set K, there exists a deterministic constant $L_K < \infty$, independent of n, such that, almost surely,*

$$\left\| \nabla_x \widehat{\ell}_n(x, t) - \nabla_x \widehat{\ell}_n(y, t) \right\| \leq L_K \|x - y\|, \quad x, y \in K, \quad n \geq 1. \tag{70}$$

Then, for every x, the input gradient of the estimated log-density ratio is a pointwise consistent estimator of the score difference:

$$\nabla_x \widehat{\ell}_n(x, t) \xrightarrow{\mathbb{P}} \nabla_x \ell(x, t) = \nabla_x \log q_{\theta,t}(x) - \nabla_x \log p_t(x). \tag{71}$$

Proof. For notation simplicity, we omit t in the notation. Fix x and a coordinate unit vector e_j. Choose a compact ball K containing x in its interior, and take $h > 0$ sufficiently small that $x + se_j \in K$ for all $s \in [0, h]$. Define the finite difference along direction e_j:

$$D_{h,j} f(x) := \frac{f(x + he_j) - f(x)}{h}. \tag{72}$$

Let ∇_j be the directional derivative along e_j. By the fundamental theorem of calculus, we have

$$D_{h,j}\widehat{\ell}_n(x) - \nabla_j \widehat{\ell}_n(x) = \frac{1}{h}\int_0^h \left(\nabla\widehat{\ell}_n(x+se_j) - \nabla\widehat{\ell}_n(x)\right)^\top e_j \, ds. \tag{73}$$

Since $\|e_j\| = 1$, the uniform local Lipschitz condition implies almost surely we have

$$\left\|D_{h,j}\widehat{\ell}_n(x) - \nabla_j \widehat{\ell}_n(x)\right\| \le \frac{1}{h}\int_0^h L_K s \, ds = \frac{L_K h}{2} \tag{74}$$

Applying the triangle inequality therefore gives

$$\left\|\nabla_j\widehat{\ell}_n(x) - \nabla_j \ell(x)\right\| \le \left\|\nabla_j\widehat{\ell}_n(x) - D_{h,j}\widehat{\ell}_n(x)\right\| + \left\|D_{h,j}\widehat{\ell}_n(x) - \nabla_j\ell(x)\right\| \tag{75}$$

$$\le \frac{L_K h}{2} + \left\|D_{h,j}\widehat{\ell}_n(x) - D_{h,j}\ell(x)\right\| + \|D_{h,j}\ell(x) - \nabla_j\ell(x)\| \tag{76}$$

$$\le \frac{L_K h}{2} + \frac{\left|\widehat{\ell}_n(x+he_j) - \ell(x+he_j)\right| + \left|\widehat{\ell}_n(x) - \ell(x)\right|}{h} + \|D_{h,j}\ell(x) - \nabla_j\ell(x)\|. \tag{77}$$

Let $\varepsilon > 0$. Because ℓ is differentiable and L_K is independent of n, we can first choose a fixed $h > 0$ sufficiently small that the sum of the first and last terms is less than $\varepsilon/2$. For this fixed h, the middle term converges to zero in probability by pointwise consistency at x and $x + he_j$. Hence,

$$\mathbb{P}\Big(\left|\nabla_j\widehat{\ell}_n(x) - \nabla_j\ell(x)\right| > \varepsilon\Big) \longrightarrow 0. \tag{78}$$

Applying this argument to each of the finitely many coordinates $j = 1, \ldots, d$ proves

$$\nabla\widehat{\ell}_n(x) \xrightarrow{\mathbb{P}} \nabla\ell(x). \tag{79}$$

This completes the proof. □

Role of Gaussian diffusion and scope of the result. Support overlap is a potential obstacle to density-ratio estimation for implicit generators: the original distributions q_θ and p_d may be singular or have disjoint supports. Gaussian diffusion removes this obstacle at every fixed noise level with strictly positive noise variance. Specifically, for a forward kernel $x_t = a_t x_0 + \sigma_t \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ and $\sigma_t > 0$, the noised densities are

$$p_t(x) = \int \varphi_{\sigma_t}(x - a_t u)\, p_d(du), \qquad q_{\theta,t}(x) = \int \varphi_{\sigma_t}(x - a_t u)\, q_\theta(du), \tag{80}$$

where φ_{σ_t} is the density of $\mathcal{N}(0, \sigma_t^2 I)$ and p_d, q_θ here denote the original clean probability measures. Both noised densities are strictly positive and smooth on $\mathbb{R}^d$, even when the original distributions do not admit densities. Their log-density ratio is therefore smooth, and the support condition required for density-ratio estimation is satisfied.

Thus, performing ratio estimation at $t > 0$ removes the need to assume support overlap at $t = 0$. Statistical consistency still requires the appropriate estimation assumptions, and gradient consistency additionally requires the regularity condition in Theorem C.1. In particular, Gaussian smoothing of the target distributions does not automatically impose this condition on the learned estimators.

D Detailed Setup of Toy Experiment

This section provides the detailed experimental setup for the gradient bias study in Section 3.2.

Data Distribution. The data distribution $p_d(x_0)$ is a mixture of Gaussians with 40 components in 2D. Component means are sampled uniformly in $[-40, 40]^2$ and variances are sampled log-uniformly, providing a multi-modal distribution with varying local densities. This setup allows closed-form computation of the true score $\nabla_{x_t} \log p_d(x_t)$ at any noise level t.

One-Step Diffusion Model. The one-step diffusion model $q_\theta(x_0)$ is an implicit generative model parameterized by a 4-layer MLP with hidden dimension 400 and SiLU activations, conditioned on

the noise level t. The model has been pre-trained to approximate p_d. Since q_θ lacks a closed-form density, we estimate its score using kernel density estimation (KDE) with 10,000 generated samples and bandwidth selected via Scott's rule. This provides the ground-truth score difference:

$$\Delta s_t^* \equiv \Delta s^*(x_t, t) = \nabla_{x_t} \log q_\theta(x_t) - \nabla_{x_t} \log p_d(x_t). \tag{81}$$

Methods Compared. We compare the bias of the two gradient estimators, VSD and DiffRatio.

1. **DiffRatio (ours)**: A time-conditioned classifier $c_\eta(x_t, t)$ is trained to distinguish between samples from $q_\theta(x_t)$ and $p_d(x_t)$. The classifier is a 4-layer MLP with hidden dimension 400 and SiLU activations, taking concatenated (x_t, t) as input and outputting a scalar logit. The score difference is obtained via Eq. 14: $\widehat{\Delta s_t} = \nabla_{x_t} \operatorname{logit}(1 - c_\eta(x_t, t))$.
2. **VSD**: Two separate score networks $s_{\psi_2}^{q_\theta}(x_t, t)$ and $s_{\psi_1}^{p_d}(x_t, t)$ are trained independently using DSM (Eqs. 9 and 10), and the score difference is computed as their difference $\widehat{\Delta s_t} = s_{\psi_2}^{q_\theta}(x_t, t) - s_{\psi_1}^{p_d}(x_t, t)$. Both networks use the same architecture as the classifier.

Training Details. All networks are trained for 10,000 iterations with batch size 1024 using Adam optimizer with learning rate 10^{-4}. Noise levels are sampled from a power schedule $t \sim t_{\min} + u^{1.5} \cdot (t_{\max} - t_{\min})$ where $u \sim \mathcal{U}(0, 1)$, $t_{\min} = 0.1$, and $t_{\max} = 20.0$. The classifier is trained with binary cross-entropy loss using softplus for numerical stability.

Evaluation. We evaluate each method by comparing estimated score differences against the KDE-based ground truth at 10 logarithmically-spaced noise levels in $[0.1, 20]$. At each noise level, we sample 1,000 evaluation points from the noised student distribution. We report the cosine similarity $\langle \widehat{\Delta s_t}, \Delta s_t^* \rangle / (\|\widehat{\Delta s_t}\| \|\Delta s_t^*\|)$, measuring directional alignment of the gradient estimates.

E Detailed Setup of Image Generation Experiments

This section provides the detailed experimental setup for the image generation experiments in Section 5.

We evaluate the performance of our proposed method by training one-step image diffusion models on two standard benchmarks: (1) unconditional generation on the CIFAR-10 (32×32) dataset [29], (2) class-label-conditioned generation on the ImageNet (64×64 and 512×512) dataset [10].

Our one-step diffusion model adopts the same neural network architecture as the pre-trained score model. Our implementation builds upon the EDM and EDM2 codebases [25, 26]. Specifically, we employ the EDM U-Net model architecture [25] for CIFAR-10 and ImageNet 64×64, and use the EDM2-M U-Net model architecture [26] for ImageNet 512×512. We use the EDM preconditioning for the inputs. We choose not to use the EDM output preconditioning as we are training a one-step generator which outputs images that differ significantly from the input noise.

Our density-ratio estimator $c_\eta(x_t, t)$ is implemented using the encoder portion of the teacher's U-Net architecture with a sigmoid function attached to the end to produce a scalar output between 0 and 1. This network is approximately half the size of the full U-Net used for score estimation, leading to reduced size of auxiliary networks. The one-step generator $g_\theta(z)$ is initialized using the weights from the pre-trained score model with the diffusion time argument t_{init}=2.5 fixed throughout training and sampling.

We consider three training criteria in our framework (DiKL/DiJS/DiRM) on CIFAR-10, and use the best-performing criterion (DiJS) for higher-dimensional ImageNet experiments (64×64 and 512×512). We use the variance-exploding (VE) noise schedule to define the diffusive divergences with $\sigma(t_{\text{min}}) = 0.002$ and $\sigma(t_{\text{max}}) = 80$. We follow standard hyperparameter settings for training generative models on CIFAR-10 and ImageNet as detailed in [25, 26]. We use a batch size of 64 for CIFAR-10 and 4,096 for ImageNet. The weighting function $w(t)$=σ_t^2 is used for for both datasets. For CIFAR-10 and ImageNet 64×64, we use a fixed learning rate of 5×10^{-5} for both generator and density ratio network. For ImageNet 512×512, we use a fixed learning rate of 10^{-4} for the density ratio network; for the generator we use a learning rate of 10^{-4} at the beginning and reduce the learning rate by a factor of 10 when the FID does not improve for 500 ticks. We adopt a single-step update strategy for the density-ratio estimator throughout our experiments, consistent with previous works [36, 73]. All experiments are conducted on 8 NVIDIA H100 80GB GPUs.

F Additional Image Generation Results



Figure 6: Visualization of samples from DiffRatio-DiJS trained on CIFAR10 (FID=2.39, IS=9.93).



Figure 7: Visualization of samples from DiffRatio-DiJS trained on ImageNet 64×64 (FID=1.54).

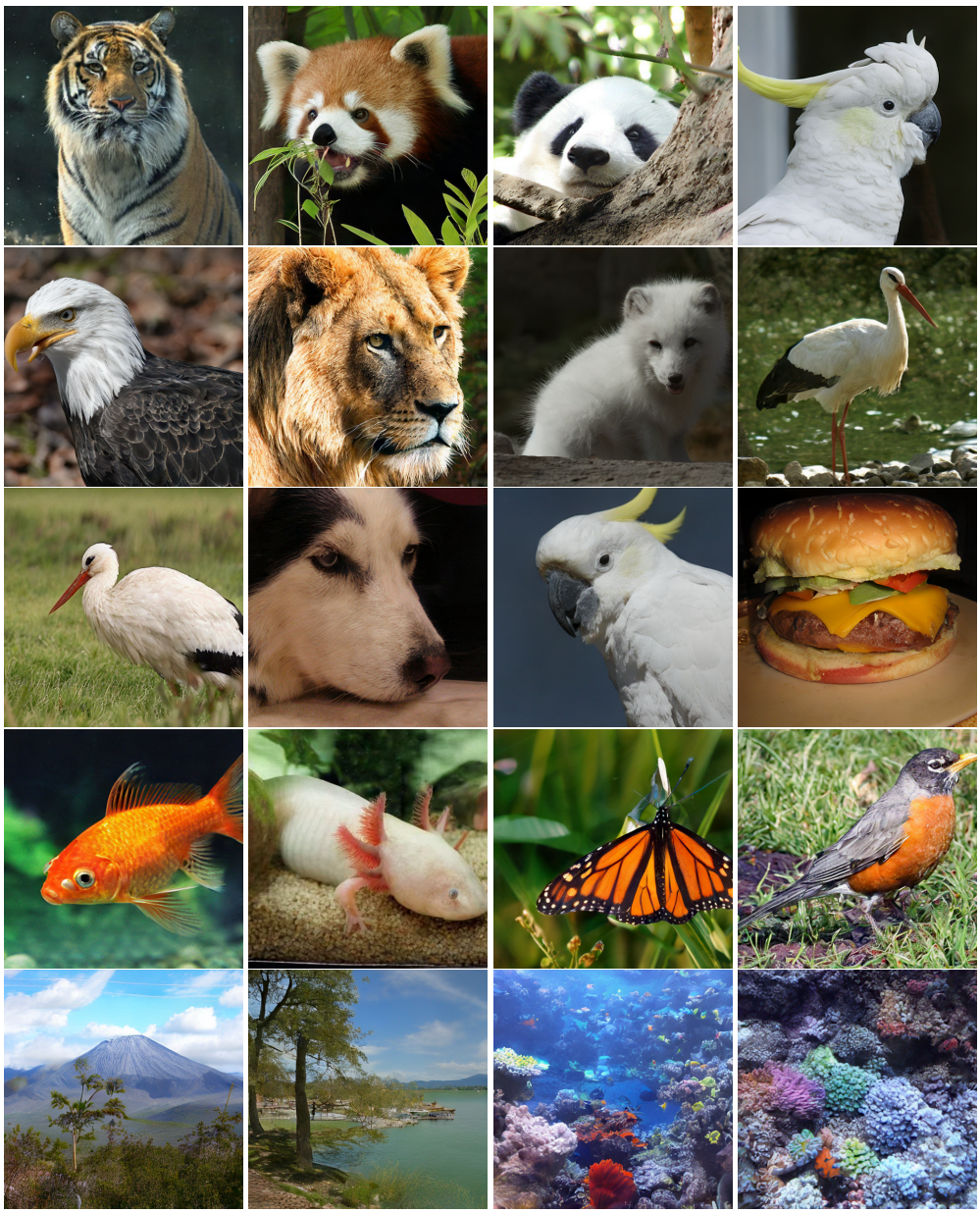


Figure 8: Visualization of samples from DiffRatio-DiJS-M trained on ImageNet 512×512 (FID=1.41).

Algorithm 2 Score-based distillation for training one-step diffusion models with teacher supervision (variational score distillation, VSD)

Require: Training samples $\mathcal{D} \sim p_d(x_0)$

// Stage 1: Pre-train a teacher score model (requires training data)

Pre-train the teacher model's score network $s^{p_d}_{\psi_1}(x_t, t)$ using DSM (Eq. 9) until convergence

// Stage 2: Distill a one-step student model with teacher supervision

Initialize the one-step generator with the teacher's score network $g_{\theta_{\text{init}}}(\cdot) \equiv s^{p_d}_{\psi_1}(\cdot, t = t_{\text{init}})$

for each training iteration **do**

    Sample time t' for student score estimation

    Estimate the student model's score by training a score network $s^{q_\theta}_{\psi_2}(x_{t'}, t')$ using DSM

    Update the one-step generator g_θ with $s^{q_\theta}_{\psi_2}(x_{t'}, t')$ (student score) and $s^{p_d}_{\psi_1}(x_{t'}, t')$ (teacher supervision) using Eq. 11

end for

G Additional Details of Variational Score Distillation

The two-stage distillation procedure of VSD is summarized in Algorithm 2.

H Limitations and Future Work

This work takes us one step closer towards training one-step diffusion models without distillation by removing teacher supervision in the training objective. However, our proposed DiffRatio approach still requires a pre-trained score network for initializing the one-step diffusion model. A key insight from our analysis in Section 6 was that while teacher supervision could be removed, weight initialization from the teacher remained essential. We found that this was not due to improved latent-to-output mappings, but rather the multi-level features learned across noise levels during diffusion training, which helped prevent mode collapse. Future work may explore unsupervised or self-supervised pretraining to provide feature-rich initializations without requiring a pre-trained diffusion model.

The inference-time parallel scaling scheme proposed in this paper allows us to use the learned density ratio network at diffusion timestep $t = 0$ to further improve the generation quality without external reward or extra sequential computation. However, density ratio estimation may become more accurate as the diffusion timestep t increases [28, 40]. It would be interesting to investigate how to leverage larger diffusion timesteps to produce a more accurate density ratio estimation for the importance sampling procedure in the proposed inference-time parallel scaling scheme.

Also, extending our framework to other modalities such as one-step diffusion models for text-to-image and text-to-video remains a promising future work direction.

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: The claims made in the abstract and introduction accurately reflect the paper's contributions and scope in Sections 3 and 5.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Limitations and future work are discussed in Appendix H.

   Guidelines:

   - The answer [N/A] 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: The main paper does not contain any theoretical results. The theoretical results in Appendices A, B and C are supported by the full set of assumptions and a complete and correct proof.

Guidelines:

- The answer [N/A] 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: The information needed to reproduce the main experimental results is provided in Section 5 and Appendix E.

   Guidelines:

   - The answer [N/A] 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: [No]

Justification: The code and model checkpoints will be made available upon the publication of this work.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental setups are provided in Section 5 and Appendix E.

   Guidelines:

   - The answer [N/A] 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: [No]

   Justification: Modern image generative models are expensive to train, test and evaluate, and the results are of relatively small variance. We follow the convention of the literature and do not report error bars.

   Guidelines:

   - The answer [N/A] 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: Information on the computer resources is provided in Appendix E.

   Guidelines:

   - The answer [N/A] 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: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: This paper advances machine learning methodologies and has no societal impacts we believe require specific discussion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] 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: The assets used in this paper are all open-sourced and properly cited.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.