Simple Distillation for One-Step Diffusion Models

Huaisheng Zhu[1], **Teng Xiao**[3,4], **Shijie Zhou**[2],
Zhimeng Guo[1], **Hangfan Zhang**[1], **Siyuan Xu**[1], **Vasant Honavar**[1]
[1]Pennsylvania State University, [2] University at Buffalo
[3] Allen Institute for AI (AI2) [4] University of Washington
hvz5312@psu.edu

Abstract

Diffusion models have established themselves as leading techniques for image generation. However, their reliance on an iterative denoising process results in slow sampling speeds, which limits their applicability to interactive and creative applications. An approach to overcoming this limitation involves distilling multistep diffusion models into efficient one-step generators. However, existing distillation methods typically suffer performance degradation or require complex iterative training procedures which increase their complexity and computational cost. In this paper, we propose Contrastive Energy Distillation (CED), a simple yet effective approach to distill multistep diffusion models into effective one-step generators. Our key innovation is the introduction of an unnormalized joint energy-based model (EBM) that represents the generator and an auxiliary score model. CED optimizes a Noise Contrastive Estimation (NCE) objective to efficiently transfers knowledge from a multistep teacher diffusion model without additional modules or iterative training complexity. We further show that CED implicitly optimizes the KL divergence between the distributions modeled by the multistep diffusion model and the one-step generator. We present results of experiments which show that CED achieves performance comparable to that of representative baselines for distilling multi-step diffusion models while maintaining excellent memory efficiency.

1 Introduction

Diffusion models have become the prominent method for image generation, capable of producing highly realistic and diverse outputs through a stable training process [18, 45, 49, 44]. Unlike Generative Adversarial Networks (GANs) [14] and Variational Autoencoders (VAEs) [26], diffusion models rely on an iterative denoising procedure that progressively refines an initial Gaussian noise into detailed images [18, 50]. This method, while effective, is computationally intensive, often requiring dozens or even hundreds of neural network evaluations. Consequently, the slow sampling speed significantly reduces the practicality of diffusion models in interactive, creative applications.

To address this issue, several efforts have aimed at reducing the sampling steps required in reverse diffusion processes [34, 36, 67, 68, 52]. These approaches still require multiple steps to generate images. To significantly improve the efficiency of diffusion models, one-step diffusion methods have been proposed. In particular, distillation techniques have become increasingly popular for one-step generation, achieving state-of-the-art performance [70]. One line of work is score-based distillation, which includes iteratively training an auxiliary score model with the one-step generator [35, 47, 61, 65] or using an additional discriminator to create a GAN [69, 64], both obtaining great results.

However, these methods heavily rely on auxiliary models and the iterative training of multiple components, leading to substantial GPU consumption, prolonged training times, and increased complexity due to the additional hyperparameters introduced by these extra components, which require careful tuning. In contrast, trajectory-based distillation offers a more lightweight alternative

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

by progressively increasing the sampling intervals of the student model [1, 15, 31, 32, 46, 51, 29]. This strategy avoids the need for extra models and iterative training, significantly simplifying the training pipeline. Nevertheless, these techniques typically experience great performance degradation compared to the original diffusion models when generating images in just one step. Therefore, in this paper, it naturally raises a question: *How can we design a simple, efficient and effective distillation algorithm for one-step diffusion models without iterative training and additional components?*

In this paper, we approach the problem by introducing an unnormalized joint Energy-based Model (EBM) that models the auxiliary score model (commonly referred to as the "fake score model" in prior work) and the one-step generator. Our key innovation lies in utilizing this EBM to distill knowledge from a teacher diffusion model. The energy function is designed to directly quantify the discrepancy between images generated by the one-step generator and outputs from the fake score model. A central challenge of our framework is to enable efficient training without relying on additional modules or complex architectures. To address this issue, we propose Contrastive Energy Distillation (CED), a method that trains our joint energy-based model inspired by the Noise Contrastive Estimation (NCE) framework [16]. Specifically, we sample synthetic images from a pre-trained teacher diffusion model prior to training, and from the generator's previous training iteration to serve as negative samples. These are then used to optimize the NCE objective. This approach removes the need for extra components and iterative updates, enabling more resource-efficient training compared to previous methods, as shown in Figure 3. Furthermore, we show that, with a specific design of the energy function, our method implicitly optimizes the Kullback–Leibler (KL) divergence—an objective widely used in distribution matching distillation of one-step diffusion models. This provides a foundation for the effectiveness of our approach in transferring knowledge from multi-step diffusion models to a single-step generator.

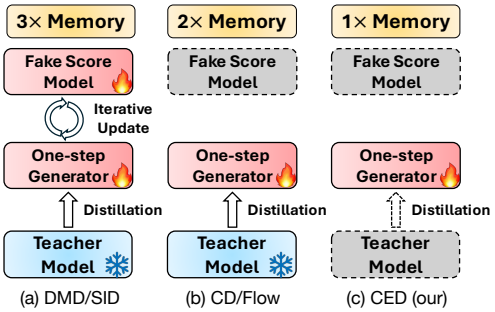

Figure 1: Training pipelines for different kinds of distillation methods. Our method (CED) uses only 1/3 or 1/2 the memory compared with them by loading fewer models during training.

In summary, the main contributions of this paper are: (i) We propose Contrastive Energy Distillation (CED), a simple and efficient distillation algorithm leveraging a contrastive loss on our joint energy-based model to transform multi-step diffusion models into a one-step generator. Inspired by noise contrastive estimation, CED eliminates the need for auxiliary modules, offering a novel and streamlined approach for designing one-step diffusion model distillation strategies. (ii) We demonstrate that CED implicitly optimizes the KL divergence for distribution matching distillation, providing a principled foundation for its distillation effectiveness. (iii) CED achieves competitive performance that also rely solely on one-step generators during training without auxiliary components.

2 Related Works

Diffusion models have become a powerful framework for generative modeling [18, 53, 50, 22], achieving remarkable success across a wide range of domains, including image generation [44, 42, 45], audio synthesis [27], video generation [10, 48], and molecular design [19, 62]. These models work by gradually transforming random noise into coherent data through a reverse diffusion process. Despite their state-of-the-art performance, the iterative nature of diffusion models leads to computational overhead, making them less suitable for real-world applications. To accelerate the sampling process of diffusion models, numerous techniques have been proposed to speed up generation through distillation. These approaches have achieved impressive results and can be broadly categorized into score-based distillation [35, 47, 61, 65, 69, 64] and trajectory-based distillation [52, 46, 32, 51, 36, 17, 63, 34, 43].

Score-based distillation methods were initially introduced in the context of text-to-3D synthesis [40, 56, 55], where a pre-trained text-to-image diffusion model is used to define a distribution matching loss. Inspired from these approaches, score distillation has been extended to the training of diffusion models themselves [65, 11, 35, 37]. These approaches either involve iteratively training an auxiliary score network alongside the one-step generator [35, 47, 61, 65], or incorporate an additional discriminator to form a GAN-like framework [69, 64] to further improve their performance. While effective, these methods depend heavily on auxiliary models and the iterative training of multiple

components, resulting in high GPU usage, longer training durations, and added complexity from extra hyperparameters associated with the additional components that require careful tuning.

Trajectory-based Distillation methods eliminate the need for additional modules and iterative training, which greatly simplify the overall pipeline. They offer a more streamlined approach by progressively increasing the sampling intervals of the student model [1, 15, 31, 32, 46, 51, 29]. Luhman et al. [34] precompute a dataset of image-noise pairs using an ODE sampler and train a student model to learn the mapping in a single step. Recified Flow simplifies the ODE trajectories to make them easier for one-step models to learn [31, 32]. Progressive Distillation eliminates the need for precomputed data by iteratively training student models, each reducing the number of sampling steps [46]. Consistency Distillation (CD) [51, 52] and TRACT [1] further improve training by enforcing self-consistency of the student's outputs along the ODE trajectory, aligning them with the teacher model. Moreover, Flow matching (FM) [31, 30] based methods distill themselves to shorten sampling steps. However, when used to generate images in a single step, these methods often suffer from significant performance degradation compared to the original diffusion models. Therefore, in this paper, we propose a simple, efficient, and effective approach to distilling a multistep diffusion model into a one-step generator using Contrastive Energy Distillation (CED). CED requires neither auxiliary models nor iterative training, and outperforms trajectory-based distillation methods, which rely on loading the teacher model during training. CED relies solely on the one-step generator during training, which make the process both simple and efficient.

3 Notations and Preliminaries

Diffusion Models. Given a dataset $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\right\}$ sampled from a real data distribution $p_{\text{real}}\left(\mathbf{x}\right)$, diffusion models learn to generate these samples by progressively denoising corrupted versions [18, 50]. In the forward diffusion process, noise is incrementally added to a sample $\mathbf{x} \sim p_{\text{real}}$ across T timesteps until it becomes pure Gaussian noise. Formally, at each timestep t, the diffused samples follow $p_{\text{real},t}\left(\mathbf{x}_t\right) = \int p_{\text{real}}\left(\mathbf{x}\right) q\left(\mathbf{x}_t \mid \mathbf{x}\right) d\mathbf{x}$ with $q(\mathbf{x}_t \mid \mathbf{x}) \sim \mathcal{N}\left(\alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I}\right)$, where $\alpha_t, \sigma_t > 0$ are scalars determined by the noise schedule. During training, the model learns to reverse this corruption process by the distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}\left(\mathbf{x}_{t-1}; \mu\left(\mathbf{x}_t, t\right), \sigma_t^2 \mathbf{I}\right)$, which predict a denoised estimate, $\mu(\mathbf{x}_t, t)$, conditioned on the timestep t and the noisy sample $\mathbf{x}_t$. By iterative refinement, the model recovers images resembling those drawn from p_{real}. Once trained, the denoised estimate aligns with the gradient of the likelihood function (i.e., the score function) of the diffused distribution:

$$s_{\text{real}}\left(\mathbf{x}_t, t\right) = \nabla_{\mathbf{x}_t} \log p_{\text{real},t}\left(\mathbf{x}_t\right) = -\frac{\mathbf{x}_t - \alpha_t \mu_{\text{real}}\left(\mathbf{x}_t, t\right)}{\sigma_t^2}. \tag{1}$$

Distribution Matching Distillation (DMD). To distill a pretrained multi-step diffusion model into a single-step generator G_θ, a distribution matching approach can be used [64, 65]. This method seeks to minimize the expected approximate Kullback–Leibler (KL) divergence, $\mathcal{L}_{\text{DMD}} = \mathbb{E}_t\left(\mathbb{D}_{\text{KL}}\left(p_{\text{fake},t} \| p_{\text{real},t}\right)\right)$, between the diffused target distribution, $p_{\text{real},t}(\mathbf{x})$, and the diffused generator distribution, $p_{\text{fake},t}$, across different timesteps t with the following objective function:

$$\nabla_\theta \mathcal{L}_{\text{DMD}} = -\mathbb{E}_t\left(\int \left(s_{\text{real}}\left(F\left(G_\theta(\mathbf{z}), t\right), t\right) - s_{\text{fake}}\left(F\left(G_\theta(\mathbf{z}), t\right), t\right)\right) \frac{dG_\theta(\mathbf{z})}{d\theta} d\mathbf{z}\right), \tag{2}$$

where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is the guassian noise, F is the diffusion forward process with noise level corresponding to time step t and s_{real} and s_{fake} are score function defined in Equation (1). DMD employs a frozen pre-trained diffusion model, μ_{real} as the teacher, while iteratively updating μ_{fake} during the training of G_θ.This involves a denoising score-matching loss applied to samples produced by the one-step generator $G_\theta(\mathbf{z})$, where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is a guassian noise.

Reinforcement Learning. When fine-tuning models with reinforcement learning, the learned reward function serves as a feedback to guide the model's training. Specifically, given a noise vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, we generate an image based on this input. Following a widely adopted optimization objective from the fine-tuning of large language models (LLMs) [20, 21], the training objective is:

$$\max_\theta \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0,\mathbf{I}), \mathbf{x} \sim G_\theta(\mathbf{x}|\mathbf{z})}\left[r_\phi(\mathbf{x}, \mathbf{z})\right] - \beta \mathbb{D}_{\text{KL}}\left[G_\theta(\mathbf{x} \mid \mathbf{z}) \| G_{\theta'}(\mathbf{x} \mid \mathbf{z})\right], \tag{3}$$

where we consider modeling the reinforcement learning process for the one-step generator G_θ and θ' serves as a reference model from a previous optimization step or a supervised fine-tuned model, similar to those commonly used in large language model literature. The reward function $r_\phi(\cdot)$ is trained prior to the reinforcement learning stage and provides feedback as the reward model.

4 Simple Distillation Approach for One-Step Diffusion Models

In this section, we introduce a simple Contrastive Energy Distillation (CED) based approach to training a one-step generator that mimics the generative process of a multistep diffusion model. We introduce a joint energy-based model that represents the auxiliary (fake) score model s_{fake} and the one-step generator G_θ. To simplify the training pipeline and avoid iterative updates involving auxiliary models (fake score models), we employ CED to directly train the one-step generator using an approach inspired by Noise Contrastive Estimation (NCE). We leverage a connection between CED and distribution matching distillation methods to design a simple and efficient method to distill a multi-step diffusion model into a one-step generator. Our approach requires loading the student model only during training, significantly reducing computational complexity and memory use while achieving competitive performance relative to state-of-the-art methods.

4.1 Joint Energy-based Models

We can interpret the two-step learning process of one-step diffusion models as leveraging a "fake score model," s_{fake} , designed to approximate the distribution of the one-step generator. However, during iterative training, optimization errors may prevent s_{fake} from accurately reflecting the true distribution of the generator. Conceptually, this discrepancy can be viewed as an energy function that captures the difference between samples generated by the fake score model and those from the one-step generator. Based on this viewpoint, we introduce an unnormalized density to simultaneously represent the fake score model's distribution and its deviation from the one-step generator.

$$p_{\theta,\phi}\left(\mathbf{x} \mid \mathbf{z}\right) = p_{\text{fake},\phi}\left(\mathbf{x} \mid \mathbf{z}\right) \frac{\exp\left(-\boldsymbol{E}\left(\mathbf{x}, G_\theta(\mathbf{z})\right)\right)}{Z_\phi\left(\mathbf{z}\right)}, \tag{4}$$

where Z_ϕ is the partition function as a normalizing factor. The term $p_{\theta,\phi}\left(\mathbf{x} \mid \mathbf{z}\right)$ denotes the distribution that generates samples $\mathbf{x}$ from an input $\mathbf{z}$ drawn from pure Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. Similarly, $p_{\text{fake},\theta}\left(\mathbf{x} \mid \mathbf{z}\right)$ denotes the distribution of samples produced by iteratively denoising $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ by the reverse process of diffusion model based on the fake score model $s_{\text{fake},\theta}$ and $\mathbf{x}$. $\boldsymbol{E}$ is used to measure the discrepancy between generated samples from fake score models and one-step generators. Moreover, $\boldsymbol{E}(\mathbf{x}, G_\theta(\mathbf{z}))$ can be viewed as a distance function between samples produced by the fake score models and those generated by the one-step generator—using, for instance, the L2, Pseudo-Huber, or LPIPS-Huber distance [6]. In this formulation, a higher energy value indicates a larger discrepancy between the two sets of samples, while a lower energy implies a lower discrepancy. Therefore, this joint EBM provides a calibrated distribution for p_{fake}: samples that are well-aligned with the one-step generator are assigned higher probabilities, while poorly aligned samples receive lower probabilities. In other words, $p_{\theta,\phi}$ offers a more accurate estimation of the image distribution of G_θ, and also capturing the interplay between the fake score model and the one-step generator.

4.2 Training with Contrastive Energy Distillation

The goal of distillation methods is to align the distribution of the fake score model (p_{fake}) with that of the teacher diffusion model (p_{real}). In this paper, we use the calibrated distribution (joint EBM) $p_{\theta,\phi}$ introduced in earlier sections, to approximate the teacher model p_{real}. Accordingly, we treat the joint EBM defined in Equation (4) as modeling the teacher distribution p_{real}. Based on this formulation, we optimize the parameters of the joint energy function using a conditional variant of Noise Contrastive Estimation (NCE) [16]. First, NCE samples contrastive examples from both the data distribution and a noise distribution that should closely approximate the data distribution. Second, it computes the likelihoods of these samples under both the model and the noise distributions, then optimizing a binary classification objective. Under our joint energy formulation from Equation (1), the log-odds reduce to $\log p_{\theta,\phi} - \log p_\phi = -\boldsymbol{E}(\mathbf{x}, G_\theta(\mathbf{z}))$, simplifying the objective to a standard binary classification form with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ and referred to Contrastive Energy Distillation (CED):

$$\mathcal{L}_{\text{CED}} = -\mathbb{E}_{\mathbf{x}\sim p_{\theta,\phi}}[\log \frac{1}{1+\exp\left(\boldsymbol{E}\left(\mathbf{x}, G_\theta(\mathbf{z})\right)\right)}] - \mathbb{E}_{\mathbf{x}'\sim p_\phi}[\log \frac{1}{1+\exp\left(-\boldsymbol{E}\left(\mathbf{x}', G_\theta(\mathbf{z})\right)\right)}]. \tag{5}$$

In this formulation, we replace $p_{\theta,\phi}$ with the teacher model $p_{\text{real}}(\mathbf{x} \mid \mathbf{z})$ as our target distilled distribution for sampling positive examples in the first term. Meanwhile, one-step diffusion distillation typically involves fitting the fake score model p_ϕ to outputs from the generator $G_{\theta'}(\mathbf{z})$ of the previous

Algorithm 1 Simple Distillation of One-step Diffusion Models

Require: Teacher diffusion model p_{true}
1: Initialize the one-step generator with the teacher's score network $G_{\theta_{\text{init}}}(\cdot) \equiv s_{\text{true}}(\cdot)$
2: Sample $\mathbf{x}$ from $p_{\text{true}}(\mathbf{x} \mid \mathbf{z})$ with $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
3: Warm-up train with $\mathcal{L} = \boldsymbol{E}(\mathbf{x}, \mathbf{z})$ to obtain a ones-step generator $G_{\theta'}$
4: Get negative samples $\mathbf{x}'$ from $G_{\theta'}(\mathbf{z})$ and construct a triplet $(\mathbf{x}, \mathbf{x}', \mathbf{z})$ for each $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
5: **for** each training iteration **do**
6: Get a batch of samples $(\mathbf{x}, \mathbf{x}', \mathbf{z})$
7: Update one-step generator's parameters θ with $\mathcal{L}_{\text{CED}}$ in Equation (14) with $(\mathbf{x}, \mathbf{x}', \mathbf{z})$
8: **end for**

optimization step, where θ' denotes the earlier parameter set of θ. To streamline the iterative process and avoid training an additional fake-score model, we replace p_ϕ with $G_{\theta'}$ for the negative samples in the second term of Equation (5). Since p_ϕ is designed to approximate the distribution of images from the previous optimization step—following earlier iterative learning approaches for one-step diffusion models—this substitution remains valid. Thus, we propose the following loss to distill the teacher model without extra models or iterative training, thereby simplifying the distillation process:

$$\mathcal{L}_{\text{CED}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}}[\log\left(1 + \exp\left(\boldsymbol{E}\left(\mathbf{x}, G_\theta(\mathbf{z})\right)\right)\right)] + \mathbb{E}_{\mathbf{x}' \sim G_{\theta'}}[\log\left(1 + \exp\left(-\boldsymbol{E}\left(\mathbf{x}', G_\theta(\mathbf{z})\right)\right)\right)], \quad (6)$$

where the energy function $\boldsymbol{E}$ serves as a distance metric, where smaller values correspond to reduced discrepancy between the two input samples. For instance, when using the L2 distance, we have $\boldsymbol{E}\left(\mathbf{x}, G_\theta(\mathbf{z})\right) = \|x - G_\theta(\mathbf{z})\|_2^2$. To further streamline the training process, we begin with a brief warm-up phase for the one-step generator $G_{\theta'}$. We then optimize the objective specified in Equation (6). The full details of this procedure can be found in Algorithm 1.

4.3 Relation to Distribution Matching Distillation

In this section, we establish the connection between our proposed objective and Distribution Matching Distillation (DMD) methods. We begin by formulating the KL divergence objective of distribution matching distillation from a reinforcement learning perspective:

$$\begin{aligned}
&\max_\theta -\mathbb{D}_{\text{KL}}\left(p^\theta_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \| p_{\text{real}}(\mathbf{x} \mid \mathbf{z})\right) = \mathbb{E}_{p^\theta_{\text{fake}}}\left[\log p_{\text{real}}(\mathbf{x} \mid \mathbf{z}) - \log p_{\text{fake}}(\mathbf{x} \mid \mathbf{z})\right] \\
&\mathbb{E}_{p^\theta_{\text{fake}}}\left[\log \frac{p_{\text{real}}(\mathbf{x} \mid \mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})} - \log \frac{p^\theta_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}\right] = \mathbb{E}_{p^\theta_{\text{fake}}} r(\mathbf{x}, \mathbf{z}) - \mathbb{D}_{\text{KL}}\left(p^\theta_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \| p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})\right),
\end{aligned} \quad (7)$$

where $r(\mathbf{x}, \mathbf{z}) = \log(p_{\text{real}}(\mathbf{x} \mid \mathbf{z}) / p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}))$ can be viewed as an auxiliary reward function and θ_0 is the initial parameter of the fake score model, initialized from the teacher diffusion model. Hence, DMD can be interpreted in a manner analogous to Reinforcement Learning From Human Feedback (RLHF) [39, 41, 60, 58, 59]. Specifically, the procedure has two stages: (1) learning a reward model r, and (2) optimizing the primary objective. To relate DMD to our single-step training approach, we reformulate it so that these two steps are combined into a single unified procedure. Specifically, we begin with the standard RLHF training pipeline, where the first step is to learn a reward model. For the DMD objective in Equation (7), we use Density Ratio Estimation methods to learn the reward model $\log(p_{\text{real}}(\mathbf{x} \mid \mathbf{z}) / p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}))$. Since this process relies on estimating the log ratio, a straightforward solution is to train a classifier (i.e., a discriminator) with logistic regression to approximate it:

$$\mathcal{L}_{\text{DRE}} = -\mathbb{E}_{\mathbf{x} \sim p_{\text{real}}}[\log\left(\sigma(h(\mathbf{x}, \mathbf{z}))\right)] - \mathbb{E}_{\mathbf{x}' \sim p^{\theta_0}_{\text{fake}}}[\log\left(1 - \sigma(h(\mathbf{x}, \mathbf{z}))\right)], \quad (8)$$

where $\sigma(\cdot)$ denotes the sigmoid function, and each data sample is treated as though it is drawn from a distribution with binary labels—one class for samples from p_{real} and one class for samples from $p^{\theta_0}_{\text{fake}}$. Then, the log density ratio can be linked to the optimal classifier probabilities via Bayes' rule [2]:

$$\log \frac{p_{\text{real}}\left(\mathbf{x} \mid \mathbf{z}\right)}{p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})} = \frac{P(c=1)P(c=0 \mid \mathbf{x}, \mathbf{z})}{P(c=0)P(c=1 \mid \mathbf{x}, \mathbf{z})} = \log\left(\frac{\sigma\left(h^*(\mathbf{x}, \mathbf{z})\right)}{1 - \sigma\left(h^*(\mathbf{x}, \mathbf{z})\right)}\right), \quad (9)$$

where $h^*(\cdot)$ is the optimal solution for Equation (8) and is the constant ratio $P(c=1)/P(c=0)$ between the priors of two classes that can be estimated with sample size. Although we can optimize

Equation (7) with the learned reward (i.e., density ratio) from Equation (8), this still involves a two-step process and additional computational overhead. To simplify matters and align with our proposed objective of Equation (6), we propose a direct optimization method for the DMD objective that does not require an RL training loop or a separate discriminator. The key insight lies in using a specific parameterization for the discriminator, which allows us to extract the optimal solution in closed form—circumventing the iterative RL loop in Equation (7). Specifically, under this parameterization, the optimization problem defined in Equation (7) admits a straightforward analytical solution:

$$p_{\text{fake}}^{\theta^*}(\mathbf{x} \mid \mathbf{z}) = \frac{1}{Z(\mathbf{z})} p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z}) \exp(r(\mathbf{x}, \mathbf{z})), \tag{10}$$

where $Z(\mathbf{z}) = \int_{\mathbf{x}} p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z}) \exp(r(\mathbf{x}, \mathbf{z})) = \int_{\mathbf{x}} p_{\text{real}}(\mathbf{x} \mid \mathbf{z}) = 1$ by the definition of $r(\cdot)$ in Equation (7) and θ^* is the optimal solution of paramter θ. A detailed derivation of this optimal format (Equation (10)) is provided in the Appendix A.1. To unify the process into a single optimization step, we can combine Equations (10) and (9) with simple algebra, yielding the following relationship:

$$\log \frac{p_{\text{fake}}^{\theta^*}(\mathbf{x} \mid \mathbf{z})}{p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z})} = \log \left(\frac{\sigma\left(h^*(\mathbf{x}, \mathbf{z})\right)}{1 - \sigma\left(h^*(\mathbf{x}, \mathbf{z})\right)} \right) \Rightarrow h^*(\mathbf{x}, \mathbf{z}) = \log \frac{p_{\text{fake}}^{\theta^*}(\mathbf{x} \mid \mathbf{z})}{p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z})}. \tag{11}$$

Then, by combining the two-step optimization in Equation (8) with Equations (7) and (5), we derive a direct optimization algorithm, resulting in the following objective:

$$\mathcal{L}_{\text{DRE}}^* = -\mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \left[\log \left(\sigma(\log \frac{p_{\text{fake}}^{\theta^*}(\mathbf{x} \mid \mathbf{z})}{p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z})}) \right) \right] - \mathbb{E}_{\mathbf{x}' \sim p_{\text{fake}}^{\theta_0}} \left[\log \left(1 - \sigma(\log \frac{p_{\text{fake}}^{\theta^*}(\mathbf{x} \mid \mathbf{z})}{p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z})}) \right) \right]. \tag{12}$$

For $p_{\text{fake}}^{\theta^*}(\mathbf{x} \mid \mathbf{z})$, we can use a single-step generator parameter, modeling the generator as a Gaussian distribution $p_{\text{fake}}^{\theta} \sim \mathcal{N}(G_\theta, \sigma^2 \mathbf{I})$. Then, we can convert this objective with the trainable parameters by replacing θ^* with θ and we can see the relation with our proposed objective in Equation (6):

$$\mathcal{L}_{\text{DRE}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}}[\log\left(1 + \exp\left(\boldsymbol{E}\left(\mathbf{x}, G_\theta(\mathbf{z})\right)\right)\right)] + \mathbb{E}_{\mathbf{x}' \sim G_{\theta'}}[\log\left(1 + \exp\left(-\boldsymbol{E}\left(\mathbf{x}', G_\theta(\mathbf{z})\right)\right)\right)], \tag{13}$$

where $\boldsymbol{E}\left(\mathbf{x}, G_\theta(\mathbf{z})\right) = -(\log p_{\text{fake}}^{\theta}(\mathbf{x} \mid \mathbf{z}) - \log p_{\text{fake}}^{\theta_0}(\mathbf{x} \mid \mathbf{z})) \approx -(\|\mathbf{x} - G_\theta(\mathbf{z})\|_2^2 - \|\mathbf{x} - G_{\theta'}(\mathbf{z})\|_2^2)$. Therefore, although the objective in Equation (8) requires a two-step training process from the reinforcement learning perspective, we unify these steps by noting that they have the relation between two steps' optimal parameter (see Equations (11) and (12)). By optimizing Equation (13) to its optimal solution, we implicitly fulfill the DMD objective, effectively merging the original two-step learning procedure into a single step. This streamlined strategy aligns closely with methods commonly employed in Direct Alignment Algorithms for RLHF [41], which we further discuss in Appendix A.2. Furthermore, implicitly optimizing DMD objectives can be viewed as a special case of our proposed CED framework, realized through a specific definition of the underlying energy functions.

4.4 Generalized Extensions and Practical Implementations

We have presented a simple distillation algorithm for one-step diffusion models that aligns with distribution-matching objectives in Equation (7), and we will next explore an extension of this approach using different loss formulations. Since our CED in Equation (13) and (6) can be seen as a binary classification problem, it naturally extends to a more general form [5, 13]:

$$\mathcal{L}_{\text{CED}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} f^+(-\boldsymbol{E}(\mathbf{x}, G_\theta(\mathbf{z}))) + \mathbb{E}_{\mathbf{x}' \sim G_{\theta'}} f^-(-\boldsymbol{E}(\mathbf{x}', G_\theta(\mathbf{z}))). \tag{14}$$

We summarize the different classification loss functions f^+ and f^- (e.g., Logistic, Hinge [7], Brier [13], Exponential [12]) in Table 1, including the Logistic loss from Equation (11). For practical implementation, the complete training procedure is detailed in Algorithm 1. To simplify the training process and reduce computational costs, we first generate synthetic samples from the teacher diffusion model and perform a warm-up phase by optimizing the one-step generator to reconstruct images from pure Gaussian noise. Specifically, we minimize the discrepancy

Table 1: Summary of various functions of generalized extensions of CED for Equation (14).

Loss	$\boldsymbol{f^+(-E)}$	$\boldsymbol{f^-(-E)}$
Logistic	$\log(1 + e^{\boldsymbol{E}})$	$\log(1 + e^{-\boldsymbol{E}})$
Hinge	$\max(0, 1 + \boldsymbol{E})$	$\max(0, 1 - \boldsymbol{E})$
Brier	$\left(\frac{e^{\boldsymbol{E}}}{1+e^{\boldsymbol{E}}}\right)^2$	$\left(\frac{1}{1+e^{\boldsymbol{E}}}\right)^2$
Exponential	$e^{\boldsymbol{E}}/2$	$e^{-\boldsymbol{E}}/2$

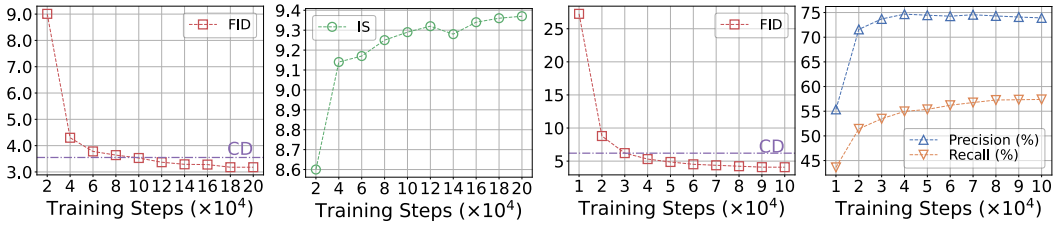

Figure 2: Results on Cifar-10 and ImageNet of different training steps, where CD [52] requires 800,000 steps to reach FID 3.55 on CIFAR-10 and 600,000 steps to get FID 6.20 on ImageNet.

with the auxiliary energy function $\boldsymbol{E}(\mathbf{z})$, obtaining an intermediate model θ', which subsequently provides negative samples for the final training objective (Line 3). To further enhance efficiency, we construct a dataset consisting of triplets $(\mathbf{x}, \mathbf{x}', \mathbf{z})$ where each noise vector $\mathbf{z}$ corresponds to positive samples $\mathbf{x}$ from the teacher diffusion model and negative samples $\mathbf{x}'$ from the warm-up generator $G_{\theta'}$ before training. This dataset with $(\mathbf{x}, \mathbf{x}', \mathbf{z})$ is then used to optimize G_θ according to Equation (14).

5 Experiment

5.1 Experimental Setup

Datasets and Models. To thoroughly evaluate CED, we use two representative benchmark datasets from prior works: CIFAR-10 (32×32) for unconditional generation [28] and ImageNet (64×64) for conditional generation [8]. To distill the teacher diffusion models, we employ EDM models [22], which are widely used in distillation-based methods.

Evaluation Protocol. We measure image generation quality using the Frechet Inception Distance (FID) and the Inception Score (IS) [38]. For FID, we generate 50k samples and compare them against the training set used by the EDM teacher model as the reference. We also consider Precision and Recall when evaluating conditional generation on ImageNet 64x64, where we follow previous works with a predefined reference batch to compute both metrics [9, 52].

Baselines. We assess the effectiveness of one-step distillation methods and other efficient diffusion models without directly utilizing the original image dataset, instead focusing on comparing the quality of images generated by CED. Based on the models employed during student model training, we categorize the baseline methods into the following groups: (1) Diffusion Models, which include methods aimed at accelerating sampling speed as well as original diffusion models; (2) (Generative Adversarial Networks) GANs with extra discriminator models; (3) Distillation with Teacher & Fake Score Models, which utilize both Teacher and Auxiliary (Fake) Score Models; (4) Distillation with Teacher Models; and (5) Distillation with Student Models Only. Note that these categorizations are based specifically on the models involved during the training of student generators.

Implementation Details. We implement CED on top of the EDM codebase [22], initializing the one-step generator θ with θ_{true} from the teacher diffusion EDM model. To reduce computational overhead in Equation (14) (see Algorithm 1), we randomly sample negative samples for only 25% of the batch size—compared to the positive samples—to optimize the objective effectively. Moreover, we compute the energy function using the LPIPS-Huber distance [51, 29], which balances perceptual similarity and robustness to outliers. As outlined in Algorithm 1, we first sample from the teacher model to generate training data. Specifically, we use 35 sampling steps for the Cifar-10 and 79 steps for ImageNet with Heun's 2nd-order method [22], as shown in Line 2 of Algorithm 1.

5.2 Image Generation

We present the image generation results comparing various baseline methods on the CIFAR-10 and ImageNet datasets in Tables 2 and 3, respectively. Specifically, we include two variants of our approach: CED-DRE and CED. The key difference between these variants is whether we load the previously obtained warm-up student model parameters θ_0 during training. From the density ratio estimation perspective, CED-DRE calculates the energy as $\boldsymbol{E}(\mathbf{x}, G_\theta(\mathbf{z})) = -(\|\mathbf{x} - G_\theta(\mathbf{z})\|_2^2 - \|\mathbf{x} - G_{\theta'}(\mathbf{z})\|_2^2)$, where we replace the traditional L2 distance with the LPIPS-Huber distance. Firstly, we observe that preserving the intermediate model parameters θ' does not significantly impact the final performance. Therefore, to further save memory, we can safely omit storing these parameters. Instead,

Table 2: Results of unconditional image generation on CIFAR-10 with FID and IS.

METHOD	NFE (↓)	FID (↓)	IS (↑)
Diffusion models			
Score SDE [53]	2000	2.38	9.83
DDPM [18]	1000	3.17	9.46
LSGM [54]	147	2.10	
EDM [22]	35	1.97	
GANs			
BigGAN [4]	1	14.7	9.22
StyleGAN2 [24]	1	8.32	9.21
StyleGAN2-ADA [23]	1	2.92	9.83
Distillation with Teacher & Fake Score Models			
Diff-Instruct [35]	1	4.53	9.89
DMD [65]	1	3.77	
SID [70]	1	1.92	9.98
Distillation with Teacher Models			
Knowledge Distillation [34]	1	9.36	
DFNO (LPIPS) [68]	1	3.78	
TRACT [1]	1	3.78	
PD [46]	1	9.12	
CD (LPIPS) [52]	1	3.55	9.48
CTM w/o GAN [25]	1	5.19	
1-rectified flow (+distill) [31]	1	6.18	9.08
2-rectified flow [31]	1	12.21	8.08
+distill [31]	1	4.85	9.01
3-rectified flow [31]	1	8.15	8.47
+Distill [31]	1	5.21	8.79
Distillation with Student Models Only			
CED-DRE	1	**2.95**	9.36
CED	1	2.96	**9.42**

Table 3: Results of conditional image generation on ImageNet 64 $\times$ 64 with FID, Precision and Recall.

METHOD	NFE (↓)	FID (↓)	Prec. (↑)	Rec. (↑)
Diffusion models				
DDIM [50]	50	13.7	0.65	0.56
	10	18.3	0.60	0.49
DPM solver [33]	10	7.93		
	20	3.42		
DEIS [66]	10	6.65		
	20	3.10		
DDPM [18]	250	11.0	0.67	0.58
iDDPM [38]	250	2.92	0.74	0.62
ADM [9]	250	2.07	0.74	0.63
EDM [22]	79	2.30		
GANs				
BigGAN-deep [4]	1	4.06	0.79	0.48
Distillation with Teacher & Fake Score Models				
Diff-Instruct [35]	1	5.57		
DMD [65]	1	2.62		
SID [70]	1	1.52	0.74	0.63
Distillation with Teacher Models				
TRACT [1]	1	7.43		
BOOT [15]	1	16.3	0.68	0.36
PD [46]	2	15.39	0.59	0.62
CD (LPIPS) [52]	1	6.20	0.68	0.6
CTM w/o GAN [25]	2	5.8		
Distillation with Student Models Only				
CED-DRE	1	**3.88**	0.72	**0.60**
CED	1	4.06	**0.74**	0.58

we pre-sample negative examples and directly load the current student model itself during training, as outlined in Algorithm 1. Furthermore, our results consistently demonstrate that CED outperforms baseline methods using only teacher models like Consistency Models (CM) and Flow Matching (FM), while significantly reducing memory consumption of loading models by approximately 50% during training. Also, our method CED can outperform DMD which includes iterative training with extra models on CIFAR-10 datasets. Although our method does not surpass approaches involving both teacher and auxiliary (fake) score models with iterative training like DMD and SiD on ImageNet, it simplifies the training pipeline by eliminating the necessity of loading and maintaining three models simultaneously, thus achieving a 66% reduction in memory usage of loading models. Also, unlike these methods—which require iterative updates and thus demand careful hyperparameter tuning for multiple models—CED avoids extra hyperparameter search, streamlining the overall training process.

Moreover, we evaluate CED at different training steps on CIFAR-10 and ImageNet, as shown in Figure 2. Remarkably, CED achieves strong performance with just 100,000 training steps on CIFAR-10 (FID 3.53) and 40,000 steps on ImageNet (FID 5.31). In comparison, CD requires 800,000 steps to reach a similar FID of 3.55 on CIFAR-10 and 600,000 steps to achieve an FID of 6.20 on ImageNet, while also requiring the loading of additional models during training (based on their official code implementation). These results further demonstrate the efficiency of our proposed algorithm, which achieves good performance with relatively few training steps. Notably, CED does not incur additional training overhead. This efficiency likely stems from the fact that CED avoids the need for extra timestep sampling like previous distillation methods (e.g., CD) during training, thereby simplifying the process and potentially enabling faster convergence.

5.3 Analysis and Ablation Study

In this section, we conduct experiments to analyze the impact of various distance metrics used in the energy function and different loss formulations together with analysis of generated imge quality, as summarized in Table 1.

Ablation Study with Different Distance Functions. We investigate the impact of different distance functions on

Table 4: Results on Cifar-10 with various distance functions for the energy $E(\cdot)$.

Method	FID (↓)	IS (↑)
L2	4.07	9.23
L1	3.78	9.23
Pseudo-Huber	3.04	9.34
LPIPS-Huber	**2.96**	**9.42**

(a) EDM (b) SiD (c) CED (our)

Figure 4: Case Study of generated images from different models trained on ImageNet.

model training. Specifically, we modify the energy function $E(\mathbf{x}, G_\theta(\mathbf{z}))$ in Equation (14) by substituting various loss functions, including L2 distance, L1 distance, Pseudo-Huber [6, 51], and LPIPS-Huber [29]. We adopt the logistic loss formulation as described in Table 1, and the corresponding results on CIFAR-10 are reported in Table 4. The Pseudo-Huber distance function, known for being less sensitive to outliers than the squared L2 distance, can potentially reduce gradient variance during training. Our results show that it consistently outperforms the standard L2 distance on Cifar-10, indicating its effectiveness. We also explore the LPIPS-Huber distance function, which encourages the model to minimize perceptual differences between generated samples and the ground truth, as demonstrated in prior works on generative modeling [29]. Among the tested metrics, the Pseudo-Huber distance function significantly outperforms L2 and other alternatives, as summarized in Table 4. Therefore, we select the LPIPS-Huber distance as the primary choice for our model.

Ablation Study with Different Loss Functions. We investigate the impact of different loss function formulations on the final objective in Equation 14. The specific formats of these loss functions are presented in Table 1. Our results show that both the Logistic and Hinge loss formulations achieve the best performance compared to other alternatives, making them strong candidates for practical use for training a one-step diffusion model. Additionally, in our implementation, we observe that Hinge loss leads to faster convergence (i.e., fewer training steps) compared to Logistic loss, making it a particularly attractive choice when aiming for competitive results with reduced training time.

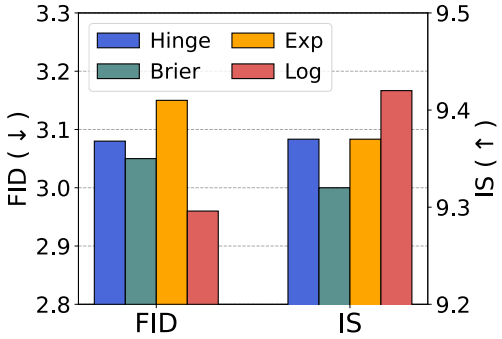

Figure 3: Results on CIFAR-10 with different loss functions, as shown in Table 1. "Exp" denotes Exponential and "Log" denotes Logistic.

Analysis of Generated Image Quality. We further generate images using different models, including EDM (our teacher model), SiD (a state-of-the-art one-step generator) and our proposed method (CED). We selected representative images from each model for qualitative comparison. Although our model (CED) does not outperform SiD in terms of FID scores, it achieves comparable image quality, as illustrated in Figure 4. Moreover, the quality of images generated by CED are also comparable to those produced by the teacher model, further demonstrating the effectiveness of our approach to generate high quality images. Notebly, CED requires only one third of the memory to load models during training, making it particularly attractive for scaling to larger models with more parameters. More examples CED-generated images are included in the Appendix C.

6 Conclusion

In conclusion, we introduce CED, a simple and efficient distillation algorithm leveraging noise contrastive estimation to distill multi-step diffusion models into a single-step generator. Our method implicitly optimizes the KL divergence commonly used for distribution-matching distillation, providing theoretical justification for its effectiveness. Empirically, CED outperforms existing one-step training methods that avoid iterative procedures but still require loading the teacher model during training, incurring additional memory overhead. In contrast, CED significantly simplifies the pipeline for one-step image generation while achieving superior performance. We believe this work paves the way for a new class of one-step distillation techniques and holds strong potential for extension to larger-scale tasks, such as text-to-image generation with high-capacity models.

Acknowledge

This work was supported in part by grants from the National Science Foundation (2226025) and the National Center for Advancing Translational Sciences and the National Institutes of Health (UL1 TR002014) and by the Center for Artificial Intelligence Foundations and Scientific Applications and the Institute for Computational and Data Sciences at Pennsylvania State University.

References

[1] 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.

[2] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9), 2009.

[3] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[5] Andreas Buja, Werner Stuetzle, and Yi Shen. Loss functions for binary class probability estimation and classification: Structure and applications. *Working draft, November*, 3:13, 2005.

[6] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311, 1997.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[8] 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.

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[10] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7346–7356, 2023.

[11] Jean-Yves Franceschi, Mike Gartrell, Ludovic Dos Santos, Thibaut Issenhuth, Emmanuel de Bézenac, Mickaël Chen, and Alain Rakotomamonjy. Unifying gans and score-based diffusion as generative particle models. *Advances in Neural Information Processing Systems*, 36:59729–59760, 2023.

[12] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[13] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[14] Ian J 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.

[15] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

[16] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models98. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

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

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

[19] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[20] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pages 1645–1654. PMLR, 2017.

[21] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

[22] 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.

[23] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33:12104–12114, 2020.

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[25] 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. *arXiv preprint arXiv:2310.02279*, 2023.

[26] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[27] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[29] Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. *Advances in Neural Information Processing Systems*, 37:63082–63109, 2024.

[30] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[31] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[32] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023.

[33] 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.

[34] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

[35] 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*, 36:76525–76546, 2023.

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

[37] Thuan Hoang Nguyen and Anh Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7807–7816, 2024.

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

[39] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[40] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[41] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[42] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[43] Yuxi Ren, Xin Xia, Yanzuo Lu, Jiacheng Zhang, Jie Wu, Pan Xie, Xing Wang, and Xuefeng Xiao. Hyper-sd: Trajectory segmented consistency model for efficient image synthesis. *arXiv preprint arXiv:2404.13686*, 2024.

[44] 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.

[45] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[46] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[47] Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching. *Advances in Neural Information Processing Systems*, 37:36046–36070, 2024.

[48] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[49] 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.

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

[51] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[52] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

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

[54] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. 2021. *URL https://arxiv. org/abs/2106.05931*, 2021.

[55] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12619–12629, 2023.

[56] 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*, 36:8406–8441, 2023.

[57] Teng Xiao, Mingxiao Li, Yige Yuan, Huaisheng Zhu, Chao Cui, and Vasant G Honavar. How to leverage demonstration data in alignment for large language model? a self-imitation learning perspective. *arXiv preprint arXiv:2410.10093*, 2024.

[58] Teng Xiao, Yige Yuan, Zhengyu Chen, Mingxiao Li, Shangsong Liang, Zhaochun Ren, and Vasant G Honavar. Simper: A minimalist approach to preference alignment without hyperparameters. *arXiv preprint arXiv:2502.00883*, 2025.

[59] Teng Xiao, Yige Yuan, Mingxiao Li, Zhengyu Chen, and Vasant G Honavar. On a connection between imitation learning and rlhf. *arXiv preprint arXiv:2503.05079*, 2025.

[60] Teng Xiao, Yige Yuan, Huaisheng Zhu, Mingxiao Li, and Vasant G Honavar. Cal-dpo: Calibrated direct preference optimization for language model alignment. *Advances in Neural Information Processing Systems*, 37:114289–114320, 2024.

[61] 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, 2024. *URL https://arxiv. org/abs/2405.16852*.

[62] Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pages 38592–38610. PMLR, 2023.

[63] Hanshu Yan, Xingchao Liu, Jiachun Pan, Jun Hao Liew, Qiang Liu, and Jiashi Feng. Perflow: Piecewise rectified flow as universal plug-and-play accelerator. *arXiv preprint arXiv:2405.07510*, 2024.

[64] 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.

[65] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.

[66] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

[67] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:49842–49869, 2023.

[68] 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.

[69] Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. *arXiv preprint arXiv:2410.14919*, 2024.

[70] 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 Omitted Derivation Details and Background

A.1 Derivation of Equation (10)

We derive Equation (9) following [41, 57]. We first consider the Distribution Matching Distillation objective in Equation (7) from the reinforcement learning perspective:

$$\max_{\theta} \mathbb{E}_{p^{\theta}_{\text{fake}}} r(\mathbf{x}, \mathbf{z}) - \beta \mathbb{D}_{\text{KL}} \left[p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \| p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \right], \tag{15}$$

where we introduce a hyperparameter β to control the weight of KL divergence. The reward function is $r(\mathbf{x}, \mathbf{z}) = \log(p_{\text{real}}(\mathbf{x} \mid \mathbf{z}) / p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}))$ and θ_0 is the parameter of the reference model. Then, we have:

$$\begin{aligned}
&\max_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I}),\mathbf{x}\sim p^{\theta}_{\text{fake}}} [r(\mathbf{x}, \mathbf{z})] - \beta \mathbb{D}_{\text{KL}} \left[p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \| p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \right] \\
=& \max_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \mathbb{E}_{p^{\theta}_{\text{fake}}} \left[r(\mathbf{x}, \mathbf{z}) - \beta \log \frac{p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})} \right] \\
=& \min_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \mathbb{E}_{p^{\theta}_{\text{fake}}} \left[\log \frac{p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})} - \frac{1}{\beta} r(\mathbf{x}, \mathbf{z}) \right] \\
=& \min_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \mathbb{E}_{p^{\theta}_{\text{fake}}} \left[\log \frac{p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}{\frac{1}{Z(\mathbf{z})} p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{z})\right)} - \log Z(\mathbf{z}) \right],
\end{aligned} \tag{16}$$

where $Z(\mathbf{z}) = \sum_{\mathbf{x}} \pi_{\text{ref}}(\mathbf{x} \mid \mathbf{z}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{z})\right)$. Then, we can obtain:

$$\begin{aligned}
&\min_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \mathbb{E}_{p^{\theta}_{\text{fake}}} \left[\log \frac{p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}{\frac{1}{Z(\mathbf{z})} p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{z})\right)} - \log Z(\mathbf{z}) \right] \\
=& \min_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \mathbb{E}_{p^{\theta}_{\text{fake}}} \left[\log \frac{p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})}{p^{\theta^*}_{\text{fake}}(\mathbf{x} \mid \mathbf{z})} - \log Z(\mathbf{z}) \right], \\
=& \min_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \left[\mathbb{D}_{\text{KL}} \left(p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \| p^{\theta^*}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \right) - \log Z(\mathbf{z}) \right]
\end{aligned} \tag{17}$$

where $p^{\theta^*}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) = \frac{1}{Z(\mathbf{z})} p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{z})\right)$. We find that the previous equation is to optimize the KL divergence and obtain the optimal parameter as follows:

$$p^{\theta}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) = p^{\theta^*}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) = \frac{1}{Z(\mathbf{z})} p^{\theta_0}_{\text{fake}}(\mathbf{x} \mid \mathbf{z}) \exp\left(\frac{1}{\beta} r(\mathbf{x}, \mathbf{z})\right) \tag{18}$$

A.2 Direct Alignment Algorithm

We follow the represenative direct alignment algorithm Typically, fine-tuning Large Language Models (LLMs) via reinforcement learning first requires training a reward model using human-preferred data pairs, denoted as $(\mathbf{x}_w, \mathbf{x}_l)$, representing human-preferred and human-dispreferred images in our setting. The reward model can be trained on image pairs with Bradley-Terry (BT) models [3]:

$$p\left(\mathbf{x}_w \succ \mathbf{x}_l \mid \mathbf{z}\right) = \frac{\exp\left(r\left(\mathbf{x}_w, \mathbf{z}\right)\right)}{\exp\left(r\left(\mathbf{x}_w, \mathbf{z}\right)\right) + \exp\left(r\left(\mathbf{x}_l, \mathbf{z}\right)\right)}. \tag{19}$$

We can obtain the reward model based on the optimal policy as follows:

$$r(\mathbf{x}, \mathbf{z}) = \beta(\log \frac{p^{\theta^*}_{\text{fake}}(\mathbf{x}, \mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x}, \mathbf{z})} - \log Z(\mathbf{z})). \tag{20}$$

Then, we can get the BT model with the format of optimal policy:

$$p^*\left(\mathbf{x}_w \succ \mathbf{x}_l \mid \mathbf{z}\right) = \frac{1}{1 + \exp\left(\beta \log \frac{p^{\theta^*}_{\text{fake}}(\mathbf{x}_l|\mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x}_l|\mathbf{z})} - \beta \log \frac{p^{\theta^*}_{\text{fake}}(\mathbf{x}_w|\mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x}_w|\mathbf{z})}\right)} \tag{21}$$

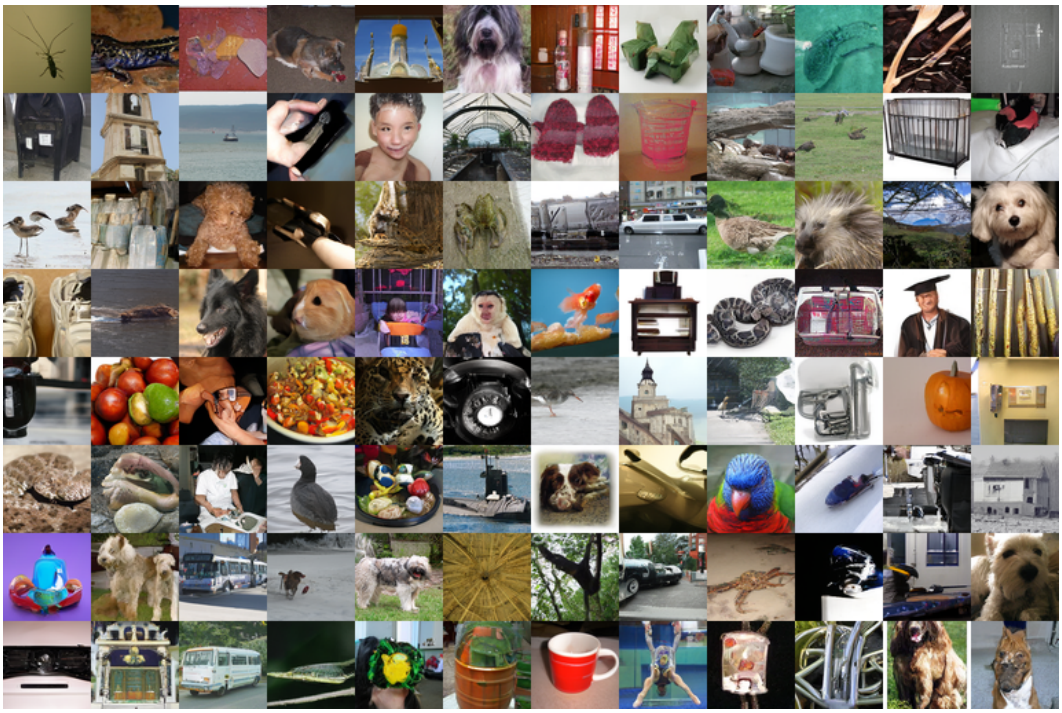

Figure 5: Images generated by one-step diffusion models trained with CED on ImageNet.

To avoid the complexity of two-step training, [41] introduces a direct optimization approach, replacing the optimal parameter θ^* with trainable parameters θ. This strategy integrates the traditional two-stage process—first training reward models and subsequently training reinforcement learning models using these reward models—into a unified, single-step training pipeline:

$$\min_{\theta} -\log p\left(\mathbf{x}_w \succ \mathbf{x}_l \mid \mathbf{z}\right) = \min_{\theta} -\log \frac{1}{1+\exp\left(\beta \log \frac{p^{\theta^*}_{\text{fake}}(\mathbf{x}_l|\mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x}_l|\mathbf{z})} - \beta \log \frac{p^{\theta^*}_{\text{fake}}(\mathbf{x}_w|\mathbf{z})}{p^{\theta_0}_{\text{fake}}(\mathbf{x}_w|\mathbf{z})}\right)}. \tag{22}$$

Therefore, our approach, viewed from the perspective of Density Ratio Estimation, shares conceptual similarities with direct alignment algorithms, as both leverage related techniques to unify the conventional two-step training process into a single step.

B Experiment Details

B.1 Implementation Details

We present implementation and setup details of CED in this section. For experiments, we use the Adam optimizer with an effective batch size of 512 for CIFAR-10 and 1024 for ImageNet. Training is conducted on 2 NVIDIA A100 GPUs. We train at fixed square resolutions and use a learning rate 3e-5. Moreover, we perform 10000 steps warm-up training as demonstrated in Line 3 of Algorithm 1. Our code of CED is based on the implementation EDM [22].

C Additional Experiments

In this section, we present more results generated from CED on Cifar-10 and ImageNet as shown in Figure 5 and Figure 6, which further shows the effectiveness of our proposed method.

Figure 6: Images generated by one-step diffusion models trained with CED on Cifar-10.

D Limitations

This work introduces an algorithm for efficiently distilling knowledge from multi-step diffusion models into one-step diffusion models for image generation tasks. However, the current focus is limited to image generation, while other domains—such as scientific discovery and 3D reconstruction, which could also benefit from fast diffusion models to enhance the efficiency of their generative pipelines.

E Broader Impact

This work can be used to accelerate both image generation and text-to-image generation, enhancing the user experience in text-to-image systems. Furthermore, it can be extended to other domains for efficient high-quality sample generation, such as 3D reconstruction, audio synthesis, and scientific discovery.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We provide the main claims in the abstract and introduction section.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations of the work are put into Appendix D.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: Our paper does not include concretely theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide the experimental results in Section 5 and code in supplementary materials to reproduce our results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide code and data in supplementary materials with sufficient instructions.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide the experimental settings in section 5 and Appendix C.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We have included statistical signifance in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We have included computing resources in Appendix B.1.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our code conforms with the NeuIPS code Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We have included broader impacts in Appendix E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have discussed this in Section 5.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

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

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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

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

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.