

# EM Distillation for One-step Diffusion Models

Sirui Xie<sup>1,2,3</sup> Zhisheng Xiao<sup>2</sup> Diederik P. Kingma<sup>1</sup> Tingbo Hou<sup>2</sup>  
 Ying Nian Wu<sup>3</sup> Kevin Murphy<sup>1</sup> Tim Salimans<sup>1</sup> Ben Poole<sup>1</sup> Ruiqi Gao<sup>1</sup>  
<sup>1</sup>Google DeepMind <sup>2</sup>Google Research <sup>3</sup>UCLA

## Abstract

While diffusion models can learn complex distributions, sampling requires a computationally expensive iterative process. Existing distillation methods enable efficient sampling, but have notable limitations, such as performance degradation with very few sampling steps, reliance on training data access, or mode-seeking optimization that may fail to capture the full distribution. We propose EM Distillation (EMD), a maximum likelihood-based approach that distills a diffusion model to a one-step generator model with minimal loss of perceptual quality. Our approach is derived through the lens of Expectation-Maximization (EM), where the generator parameters are updated using samples from the joint distribution of the diffusion teacher prior and inferred generator latents. We develop a reparametrized sampling scheme and a noise cancellation technique that together stabilize the distillation process. We further reveal an interesting connection of our method with existing methods that minimize mode-seeking KL. EMD outperforms existing one-step generative methods in terms of FID scores on ImageNet-64 and ImageNet-128, and compares favorably with prior work on distilling text-to-image diffusion models.

## 1 Introduction

Diffusion models [1–3] have enabled high-quality generation of images [4–6], videos [7, 8], and other modalities [9–11]. Diffusion models use a forward process to create a sequence of distributions that transform the complex data distribution into a Gaussian distribution, and learn the score function for each of these intermediate distributions. Sampling from a diffusion model reverses this forward process to create data from random noise by solving an SDE, or an equivalent probability flow ODE [12]. Typically, solving this differential equation requires a significant number of evaluations of the score function, resulting in a high computational cost. Reducing this cost to single function evaluation would enable applications in real-time generation.

To enable efficient sampling from diffusion models, two distinct approaches have emerged: (1) trajectory distillation methods [13–18] that accelerate solving the differential equation, and (2) distribution matching approaches [19–23] that learn implicit generators to match the marginals learned by the diffusion model. Trajectory distillation-based approaches have greatly reduced the number of steps required to produce samples, but continue to face challenges in the 1-step generation regime. Distribution matching approaches can enable the use of arbitrary generators and produce more compelling results in the 1-step regime, but often fail to capture the full distribution due to the *mode-seeking* nature of the divergences they minimize.

In this paper, we propose EM Distillation (EMD), a diffusion distillation method that minimizes an approximation of the *mode-covering* divergence between a pre-trained diffusion teacher model and a latent-variable student model. The student enables efficient generation by mapping from noise to data in just one step. To achieve Maximum Likelihood Estimation (MLE) of the marginal teacher distribution for the student, we propose a method similar to the Expectation-Maximization (EM) framework [24], which alternates between an Expectation-step (E-step) that estimates the learning gradients with Monte Carlo samples, and a Maximization-step (M-step) that updates the student

through gradient ascent. As the target distribution is represented by the pre-trained score function, the E-step in the original EM that first samples a datapoint and then infers its implied latent variable would be expensive. We introduce an alternative MCMC sampling scheme that jointly updates the data and latent pairs initialized from student samples, and develop a reparameterized approach that simplifies hyperparameter tuning and improves performance for short-run MCMC [25]. For the optimization in the M-step given these joint samples, we discover a tractable linear noise term in the learning gradient, whose removal significantly reduces variances. Additionally, we identify a connection to Variational Score Distillation [9, 26] and Diff-Instruct [22], and show how the strength of the MCMC sampling scheme can interpolate between mode-seeking and mode-covering divergences. Empirically, we first demonstrate that a special case of EMD, which is equivalent to the Diff-Instruct [22] baseline, can be readily scaled and improved to achieve strong performance. We further show that the general formulation of EMD that leverages multi-step MCMC can achieve even more competitive results. For ImageNet-64 and ImageNet-128 conditional generation, EMD outperforms existing one-step generation approaches with FID scores of 2.20 and 6.0. EMD also performs favorably on one-step text-to-image generation by distilling from Stable Diffusion models.

## 2 Preliminary

### 2.1 Diffusion models and score matching

Diffusion models [1, 2], also known as score-based generative models [27, 3], consist of a forward process that gradually injects noise to the data distribution and a reverse process that progressively denoises the observations to recover the original data distribution  $p_{\text{data}}(\mathbf{x}_0)$ . This results in a sequence of noise levels  $t \in (0, 1]$  with conditional distributions  $q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$ , whose marginals are  $q_t(\mathbf{x}_t)$ . We use a variance-preserving forward process [3, 28, 29] such that  $\sigma_t^2 = 1 - \alpha_t^2$ . Song et al. [3] showed that the reverse process can be simulated with a reverse-time Stochastic Differential Equation (SDE) that depends only on the time-dependent score function  $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$  of the marginal distribution of the noisy observations. This score function can be estimated by a neural network  $s_\phi(\mathbf{x}_t, t)$  through (weighted) denoising score matching [30, 31]:

$$\mathcal{J}(\phi) = \mathbb{E}_{p_{\text{data}}(\mathbf{x}_0), p(t), q_t(\mathbf{x}_t|\mathbf{x}_0)} [w(t) \|s_\phi(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)\|_2^2], \quad (1)$$

where  $w(t)$  is the weighting function and  $p(t)$  is the noise schedule.

### 2.2 MCMC with Langevin dynamics

While solving the reverse-time SDE results in a sampling process that traverses noise levels, simulating Langevin dynamics [32] results in a sampler that converges to and remains at the data manifold of a target distribution. As a particularly useful Markov Chain Monte Carlo (MCMC) sampling method for continuous random variables, Langevin dynamics generate samples from a target distribution  $\rho(\mathbf{x})$  by iterating through

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \gamma \nabla_{\mathbf{x}} \log \rho(\mathbf{x}^i) + \sqrt{2\gamma} \mathbf{n}, \quad (2)$$

where  $\gamma$  is the stepsize,  $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , and  $i$  indexes the sampling timestep. Langevin dynamics has been widely adopted for sampling from diffusion models [27, 3] and energy-based models [33–36]. Convergence of Langevin dynamics requires a large number of sampling steps, especially for high-dimensional data. In practice, short-run variants with early termination have been successfully used for learning of EBMs [25, 37, 38].

### 2.3 Maximum Likelihood and Expectation-Maximization

Expectation-Maximization (EM) [24] is a maximum likelihood estimation framework to learn latent variable models:  $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ , such that the marginal distribution  $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z}$  approximates the target distribution  $q(\mathbf{x})$ . It originates from the generic training objective of *maximizing* the log-likelihood function over parameters:  $\mathcal{L}(\theta) = \mathbb{E}_{q(\mathbf{x})}[\log p_\theta(\mathbf{x})]$ , which is equivalent to *minimizing* the *forward* KL divergence  $D_{\text{KL}}(q(\mathbf{x})||p_\theta(\mathbf{x}))$  [39]. Since the marginal distribution  $p_\theta(\mathbf{x})$  is usually analytically intractable, EM involves an E-step that expresses the gradients over the model parameters  $\theta$  with an expectation formula

$$\nabla_\theta \mathcal{L}(\theta) = \nabla_\theta \mathbb{E}_{q(\mathbf{x})}[\log p_\theta(\mathbf{x})] = \mathbb{E}_{q(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z})], \quad (3)$$



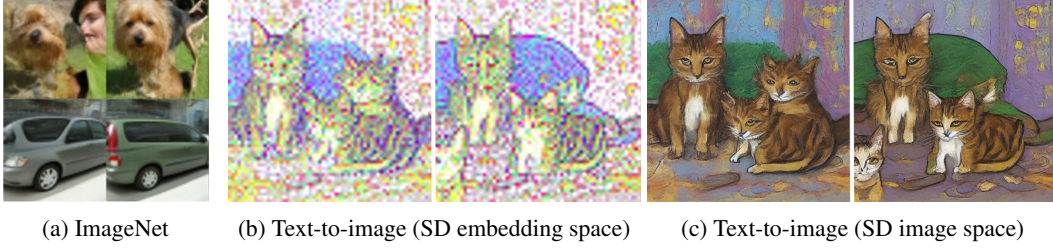


Figure 1: **Before and after MCMC correction.** In (a)(b), the left columns are  $\mathbf{x} = g_\theta(\mathbf{z})$ , the right columns are updated  $\mathbf{x}$  after 300 steps of MCMC sampling jointly on  $\mathbf{x}$  and  $\mathbf{z}$ . (a) illustrates the effect of correction in ImageNet. Note that the off-manifold images are corrected. (b) illustrates the correction in the embedding space of Stable Diffusion v1.5, which are decoded to image space in (c). Note the disentanglement of the cats and sharpness of the sofa. Zoom in for better viewing.

where  $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{x})}$  is the posterior distribution of  $\mathbf{z}$  given  $\mathbf{x}$ . See Appendix A for a detailed derivation. The expectation can be approximated by Monte Carlo samples drawn from the posterior using e.g. MCMC sampling techniques. The estimated gradients are then used in an M-step to optimize the parameters. Han et al. [40] learned generator networks with an instantiation of this EM framework where E-steps leverage Langevin dynamics for drawing samples.

## 2.4 Variational Score Distillation and Diff-Instruct

Our method is also closely related to Score Distillation Sampling (SDS) [9], Variational Score Distillation (VSD) [26] and Diff-Instruct [22], which have been used for distilling diffusion models into a single-step generator [23, 41]. The generator produces clean images  $\mathbf{x}_0 = g_\theta(\mathbf{z})$  with  $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ , and can be diffused to noise level  $t$  to form a latent variable model  $p_{\theta,t}(\mathbf{x}_t, \mathbf{z}) = p_{\theta,t}(\mathbf{x}_t|\mathbf{z})p(\mathbf{z})$ ,  $p_{\theta,t}(\mathbf{x}_t|\mathbf{z}) = \mathcal{N}(\alpha_t g_\theta(\mathbf{z}), \sigma_t^2 \mathbf{I})$ . This model is trained to match the marginal distributions  $p_{\theta,t}(\mathbf{x}_t)$  and  $q_t(\mathbf{x}_t)$  by minimizing their *reverse* KL divergence. Integrating over all noise levels, the objective is to *minimize*  $\mathcal{J}(\theta)$  where

$$\mathcal{J}(\theta) = \mathbb{E}_{p(t)}[\tilde{w}(t) D_{\text{KL}}(p_{\theta,t}(\mathbf{x}_t) || q_t(\mathbf{x}_t))] = \mathbb{E}_{p(t)} \left[ \tilde{w}(t) \int p_{\theta,t}(\mathbf{x}_t) \log \frac{p_{\theta,t}(\mathbf{x}_t)}{q_t(\mathbf{x}_t)} d\mathbf{x}_t \right]. \quad (4)$$

When parametrizing  $\mathbf{x}_t = \alpha_t g_\theta(\mathbf{z}) + \sigma_t \epsilon$ , the gradient for this objective in Eq. (4) can be written as

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{p(t), p(\epsilon), p(\mathbf{z})} \left[ -\tilde{w}(t) \left( \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{\text{teacher score}} - \underbrace{\nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t)}_{\text{learned } s_\phi(\mathbf{x}_t, t)} \right) \alpha_t \nabla_\theta g_\theta(\mathbf{z}) \right], \quad (5)$$

where  $p(\epsilon) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ , the teacher score is provided by the pre-trained diffusion model. In SDS,  $\nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t)$  is the known analytic score function of the Gaussian generator. In VSD and Diff-Instruct, an auxiliary score network  $s_\phi(\mathbf{x}_t, t)$  is learned to estimate it. The training alternates between learning the generator network  $g_\theta$  with the gradient update in Eq. (5) and learning the score network  $s_\phi$  with the denoising score matching loss in Eq. (1).

## 3 Method

### 3.1 EM Distillation

We consider formulating the problem of distilling a pre-trained diffusion model to a deep latent-variable model  $p_{\theta,t}(\mathbf{x}_t, \mathbf{z})$  defined in Section 2.4 using the EM framework introduced in Section 2.3. For simplicity, we begin with discussing the framework at a single noise level and drop the subscript  $t$ . We will revisit the integration over all noise levels in Section 3.3. Assume the target distribution  $q(\mathbf{x})$  is represented by the diffusion model where we can access the score function  $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ . Theoretically speaking, the generator network  $g_\theta(\mathbf{z})$  can employ any architecture including ones where the dimensionality of the latents differs from the data dimensionality. In this work, we reuse the diffusion denoiser parameterization as in other work on one-step distillation:  $g_\theta(\mathbf{z}) = \hat{\mathbf{x}}_\theta(\mathbf{z}, t^*)$ , where  $\hat{\mathbf{x}}_\theta$  is the  $\mathbf{x}$ -prediction function inherited from the teacher diffusion model, and  $t^*$  remains a hyper-parameter.

A naive implementation of the E-step involves two steps: (1) draw samples from the target diffusion model  $q(\mathbf{x})$  and (2) sample the latent variable  $\mathbf{z}$  from  $p_\theta(\mathbf{z}|\mathbf{x})$  with *e.g.* MCMC techniques. Both steps can be highly non-trivial and computationally expensive, so here we present an alternative approach to sampling the same target distribution that avoids directly sampling from the pretrained diffusion model, by instead running MCMC from the joint distribution of  $(\mathbf{x}, \mathbf{z})$ . We initialize this sampling process using a joint sample from the student: drawing  $\mathbf{z} \sim p(\mathbf{z})$  and  $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$ . This sampled  $\mathbf{x}$  is no longer drawn from  $q(\mathbf{x})$ , but  $\mathbf{z}$  is guaranteed to be a valid sample from the posterior  $p_\theta(\mathbf{z}|\mathbf{x})$ . We then run MCMC to correct the sampled pair towards the desired distribution:  $\rho_\theta(\mathbf{x}, \mathbf{z}) := q(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{x}, \mathbf{z}) \frac{q(\mathbf{x})}{p_\theta(\mathbf{x})}$  (see Fig. 1 for a visualization of this process). If  $q(\mathbf{x})$  and  $p_\theta(\mathbf{x})$  are close to each other,  $\rho_\theta(\mathbf{x}, \mathbf{z})$  is close to  $p_\theta(\mathbf{x}, \mathbf{z})$ . In that case, initializing the *joint* sampling of  $\rho_\theta(\mathbf{x}, \mathbf{z})$  with pairs of  $(\mathbf{x}, \mathbf{z})$  from  $p_\theta(\mathbf{x}, \mathbf{z})$  could significantly accelerate both sampling of  $\mathbf{x}$  and inference of  $\mathbf{z}$ . Assuming MCMC converges, we can use the resulting samples to estimate the learning gradients for EM:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{\rho_\theta(\mathbf{x}, \mathbf{z})} [\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z})] = \mathbb{E}_{\rho_\theta(\mathbf{x}, \mathbf{z})} \left[ -\frac{\nabla_\theta \|\mathbf{x} - \alpha g_\theta(\mathbf{z})\|_2^2}{2\sigma^2} \right]. \quad (6)$$

We abbreviate our method as EMD hereafter. To successfully learn the student network with EMD, we need to identify efficient approaches to sample from  $\rho_\theta(\mathbf{x}, \mathbf{z})$ .

### 3.2 Reparametrized sampling and noise cancellation

As an initial strategy, we consider Langevin dynamics which only requires the score functions:

$$\begin{aligned} \nabla_{\mathbf{x}} \log \rho_\theta(\mathbf{x}, \mathbf{z}) &= \underbrace{\nabla_{\mathbf{x}} \log q(\mathbf{x})}_{\text{teacher score}} - \underbrace{\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})}_{\text{learned } s_\phi(\mathbf{x})} + \underbrace{\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}|\mathbf{z})}_{-\frac{\mathbf{x} - \alpha g_\theta(\mathbf{z})}{\sigma^2}}, \\ \nabla_{\mathbf{z}} \log \rho_\theta(\mathbf{x}, \mathbf{z}) &= \nabla_{\mathbf{z}} \log p_\theta(\mathbf{x}|\mathbf{z}) + \nabla_{\mathbf{z}} \log p_\theta(\mathbf{z}) = -\frac{\mathbf{x} - \alpha g_\theta(\mathbf{z})}{\sigma^2} \alpha \nabla_{\mathbf{z}} g_\theta(\mathbf{z}) - \mathbf{z}. \end{aligned} \quad (7)$$

While we do not have access to the score of the student,  $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ , we can approximate it with a learned score network  $s_\phi$  estimated with denoising score matching as in VSD [26] and Diff-Instruct [22]. As will be covered in Section 3.3, this score network is estimated at all noise levels. The Langevin dynamics defined in Eq. (7) can therefore be simulated at any noise level.

Running Langevin MCMC is expensive and requires careful tuning, and we found this challenging in the context of diffusion model distillation where different noise levels have different optimal step sizes. We leverage a reparametrization of  $\mathbf{x}$  and  $\mathbf{z}$  to accelerate the joint MCMC sampling and simplify step size tuning, similar to Nijkamp et al. [36], Xiao et al. [42]. Specifically, the parametrization  $\mathbf{x} = \alpha g_\theta(\mathbf{z}) + \sigma \epsilon$  defines a deterministic transformation from the pair of  $(\epsilon, \mathbf{z})$  to the pair of  $(\mathbf{x}, \mathbf{z})$ , which enables us to push back the joint distribution  $\rho_\theta(\mathbf{x}, \mathbf{z})$  to the  $(\epsilon, \mathbf{z})$ -space. The reparameterized distribution is

$$\rho_\theta(\epsilon, \mathbf{z}) = \frac{q(\alpha g_\theta(\mathbf{z}) + \sigma \epsilon)}{p_\theta(\alpha g_\theta(\mathbf{z}) + \sigma \epsilon)} p(\epsilon) p(\mathbf{z}). \quad (8)$$

The score functions become

$$\begin{aligned} \nabla_\epsilon \log \rho_\theta(\epsilon, \mathbf{z}) &= \sigma(\nabla_{\mathbf{x}} \log q(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})) - \epsilon, \\ \nabla_{\mathbf{z}} \log \rho_\theta(\epsilon, \mathbf{z}) &= \alpha(\nabla_{\mathbf{x}} \log q(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})) \nabla_{\mathbf{z}} g_\theta(\mathbf{z}) - \mathbf{z}. \end{aligned} \quad (9)$$

---

#### Algorithm 1: EM Distillation

---

**Input:** Teacher score functions  $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ , generator network  $g_\theta$ , prior  $p(\mathbf{z})$ , score network  $s_\phi$ , noise scheduler  $p(t)$ , weighting functions  $w(t)$  and  $\tilde{w}(t)$ , # of MCMC steps  $K$ , MCMC step size  $\gamma$ .

**Output:** Generator network  $g_\theta$ , score network  $s_\phi$ .

**while** not converged **do**

    Sampling a batch of  $t, \mathbf{z}, \epsilon$  from  $p(t), p(\mathbf{z}), \mathcal{N}(\mathbf{0}, \mathbf{I})$  to obtain  $\mathbf{x}_t$

    Updating  $s_\phi$  via Stochastic Gradient Descent with the batch estimate of Eq. (12)

    Sampling  $\mathbf{x}_t^K$  and  $\mathbf{z}^K$  with  $(\epsilon, \mathbf{z})$ -corrector( $\mathbf{x}_0, \epsilon, \mathbf{z}, t, \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t), g_\theta, s_\phi, K, \gamma$ )

    Updating  $g_\theta$  via Stochastic Gradient Ascent with the batch estimate of Eq. (11)

**end while**

---

---

**Algorithm 2:**  $(\epsilon, \mathbf{z})$ -corrector

---

**Input:**  $\mathbf{x}_0, \epsilon, \mathbf{z}, t$ , teacher score function  $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ , generator network  $g_\theta$ , prior  $p_0(\mathbf{z})$ , score network  $s_\phi$ , # of MCMC steps  $K$ , MCMC step size  $\gamma$ .

**Output:**  $\mathbf{x}_t^K, \mathbf{z}^K$ .

Sampling Langevin noise  $\mathbf{n}^1, \mathbf{n}^2, \dots, \mathbf{n}^K$  from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , letting  $\epsilon^0 = \epsilon, \mathbf{z}^0 = \mathbf{z}$

**for**  $i$  in  $[1, K]$  **do**

    Updating  $(\epsilon^i, \mathbf{z}^i)$  with 1-step Langevin update over scores Eq. (9), with  $\epsilon^i$  updated using  $\mathbf{n}^i$

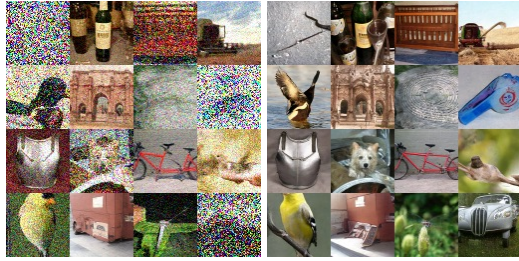
**end for**

Pushing  $(\epsilon^K, \mathbf{z}^K)$  forward to  $(\mathbf{x}_t^K, \mathbf{z}^K)$  and then canceling the noises in  $\mathbf{x}_t^K$

---

See Appendix B for a detailed derivation. We found that this parameterization admits the same step sizes across noise levels and results in better performance empirically (Table 1).

Still, learning the student with these samples continued to present challenges. When visualizing samples  $\mathbf{x}$  produced by MCMC (see Fig. 2a), we found that samples contained substantial noise. While this makes sense given the level of noise in the marginal distributions, we found that this inhibited learning of the student. We identify that, due to the structure of Langevin dynamics, there is noise added to  $\mathbf{x}$  at each step that can be linearly accumulated across iterations. By removing this accumulated noise along with the temporally decayed initial  $\epsilon$ , we recover cleaner  $\mathbf{x}$  samples (Fig. 2b). Since  $\mathbf{x}$  is effectively a regression target in Eq. (6), and the expected value of the noises is 0, canceling these noises reduces variance of the gradient without introducing bias. Empirically, we find bookkeeping the sampled noises in the MCMC chain and canceling these noises after the loop significantly stabilize the training of the generator network. This noise cancellation was critical to the success of EMD, and is detailed in Appendix B and ablated in experiments (Fig. 3ab).



(a)  $\mathbf{x}$  w/ accumulated noise (b)  $\mathbf{x}$  w/o accumulated noise

Figure 2: Images after 8-step Langevin updates with and without accumulated noise.

### 3.3 Maximum Likelihood across all noise levels

The derivation above assumes smoothing the data distribution with a single noise level. In practice, the diffusion teachers always employ multiple noise levels  $t$ , coordinated by a noise schedule  $p(t)$ . Therefore, we optimize a weighted loss over all noise levels of the diffusion model, to encourage that the marginals of the student network match the marginals of the diffusion process at all noise levels:

$$\nabla_{\theta} \mathcal{L}(\theta) = \nabla_{\theta} \mathbb{E}_{p(t), q_t(\mathbf{x}_t)} [\tilde{w}(t) \log p_{\theta, t}(\mathbf{x}_t)] = \mathbb{E}_{p(t), \rho_t(\mathbf{x}_t, \mathbf{z})} [\tilde{w}(t) \nabla_{\theta} \log p_{\theta, t}(\mathbf{x}_t, \mathbf{z})], \quad (10)$$

where  $p_{\theta, t}(\mathbf{x}_t, \mathbf{z})$  are a series of latent-variable models as defined in Section 2.4, with a shared generator  $g_\theta(\mathbf{z})$  across all noise levels. Empirically, we find  $\tilde{w}(t) = \sigma_t^2 / \alpha_t$  or  $\tilde{w}(t) = \sigma_t^2 / \alpha_t^2$  perform well.

Denote the resulted distribution after  $K$  steps of MCMC sampling with noise cancellation as  $\rho_t^K(\mathbf{x}_t^K, \mathbf{z}^K)$ , the final gradient for the generator network  $g_\theta$  is

$$\nabla_{\theta} \mathcal{L}(\theta) = \mathbb{E}_{p(t), \rho_t^K(\mathbf{x}_t^K, \mathbf{z}^K)} \left[ -\tilde{w}(t) \frac{\nabla_{\theta} \|\mathbf{x}_t^K - \alpha_t g_\theta(\mathbf{z}^K)\|_2^2}{2\sigma_t^2} \right]. \quad (11)$$

The final gradient for the score network  $s_\phi(\mathbf{x}_t, t)$  is

$$\nabla_{\phi} \mathcal{J}(\phi) = \mathbb{E}_{p(t), p_{\theta, t}(\mathbf{x}_t, \mathbf{z})} [w(t) \nabla_{\phi} \|s_\phi(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t | g_\theta(\mathbf{z}))\|_2^2]. \quad (12)$$

Similar to VSD [26, 22], we employ alternating update for the generator network  $g_\theta$  and the score network  $s_\phi(\mathbf{x}_t, t)$ . See summarization in Algorithm 1 and also Appendix C for a JAX-style implementation.

### 3.4 Connection with VSD and Diff-Instruct

In this subsection, we reveal an interesting connection between EMD and Variational Score Distillation (VSD) [26, 22], *i.e.*, although motivated by optimizing different types of divergences, VSD [26, 22] is equivalent to EMD with a special sampling scheme.

To see this, consider the 1-step EMD with noise cancellation, stepsize  $\gamma = 1$  in  $\mathbf{x}$ , and no update on  $\mathbf{z}$

$$\mathbf{x}_t^0 = \alpha_t g_{\theta}(\mathbf{z}) + \sigma_t \epsilon, \quad \mathbf{x}_t^1 = \alpha_t g_{\theta}(\mathbf{z}) + \sigma_t^2 \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}_t^0)}{p_{\theta,t}(\mathbf{x}_t^0)} \pm \sqrt{2\sigma_t} \mathbf{n}^T. \quad (13)$$

Substitute it into Eq. (11), we have

$$\begin{aligned} \nabla_{\theta} \mathcal{L}(\theta) &= \mathbb{E}_{p(t), p(\epsilon), p(\mathbf{z})} \left[ -\tilde{w}(t) \frac{\nabla_{\theta} \|\mathbf{x}_t^1 - \alpha_t g_{\theta}(\mathbf{z})\|_2^2}{2\sigma_t^2} \right] \\ &= \mathbb{E}_{p(t), p(\epsilon), p(\mathbf{z})} [\tilde{w}(t) (\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t)) \alpha_t \nabla_{\theta} g_{\theta}(\mathbf{z})], \end{aligned} \quad (14)$$

which is exactly the gradient for VSD (Eq. (5)), up to a sign difference. This insight demonstrates that, EMD framework can flexibly interpolate between mode-seeking and mode-covering divergences, by leveraging different sampling schemes from 1-step sampling in only  $\mathbf{x}$  (a likely biased sampler) to many-step joint sampling in  $(\mathbf{x}, \mathbf{z})$  (closer to a mixing sampler). Notably, for image generation, some believe that *forward* KL divergence may fail to achieve better fidelity compared to *reverse* KL divergence. The interpolation enabled by EMD can thus be very useful in practice.

If we further assume the marginal  $p_{\theta}(\mathbf{x})$  is a Gaussian, then EMD update in Eq. 14 would resemble Score Distillation Sampling (SDS) [9].

## 4 Related Work

**Diffusion acceleration.** Diffusion models have the notable issue of slowness in inference, which motivates many research efforts to accelerate the sampling process. One line of work focuses on developing numerical solvers [43, 12, 44–47] for the PF-ODE. Another line of work leverages the concept of knowledge distillation [48] to condense the sampling trajectory of PF-ODE into fewer steps [13, 49, 15, 14, 50, 51, 18, 52–55]. However, both approaches have significant limitations and have difficulty in substantially reducing the sampling steps to the single-step regime without significant loss in perceptual quality.

**Single-step diffusion models.** Recently, several methods for one-step diffusion sampling have been proposed, sharing the same goal as our approach. Some methods fine-tune the pre-trained diffusion model into a single-step generator via adversarial training [20, 21, 56], where the adversarial loss enhances the sharpness of the diffusion model’s single-step output. Adversarial training can also be combined with trajectory distillation techniques to improve performance in few or single-step regimes [52, 57, 58]. Score distillation techniques [9, 26] have been adopted to match the distribution of the one-step generator’s output with that of the teacher diffusion model, enabling single-step generation [22, 41]. Additionally, Yin et al. [23] introduces a regression loss to further enhance performance. These methods achieve more impressive 1-step generation, some of which enjoy additional merits of being data-free or flexible in the selection of generator architecture. However, they often minimize over mode-seeking divergences that can fail to capture the full distribution and therefore causes mode collapse issues. We discuss the connection between our method and this line of work in Section 3.4. Concurrent with our work, Zhou et al. [59] adopt Fisher divergence as the distillation objective and propose a novel decomposition that alleviates the dependency on the approximation accuracy of the auxiliary score network. Although the adopted Fisher divergence is similar to *reverse* KL in terms of the reparametrization and hence the risk of mode collapse, Zhou et al. [59] demonstrate impressive performance gain.

## 5 Experiments

We employ EMD to learn one-step image generators on ImageNet  $64 \times 64$ , ImageNet  $128 \times 128$  [60] and text-to-image generation. The student generators are initialized with the teacher model weights. Results are compared according to Frechet Inception Distance (FID) [61], Inception Score (IS) [62],

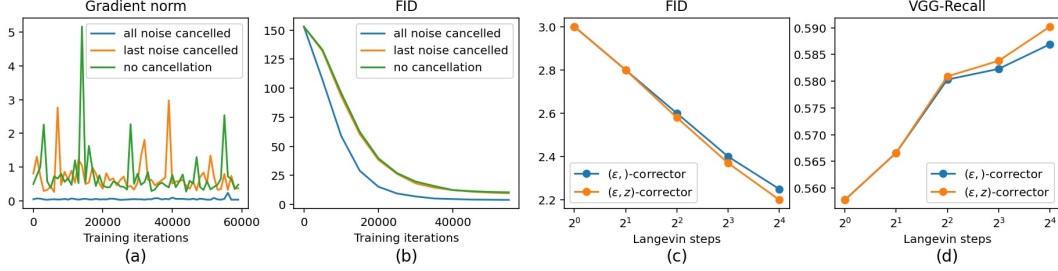


Figure 3: (a)(b) Gradient norms and FIDs for complete noise cancellation, last-step noise cancellation and no noise cancellation. (c)(d) FIDs and Recalls of EMD with different numbers of Langevin steps.

Recall (Rec.) [63] and CLIP Score [64]. Throughout this section, we will refer to the proposed EMD with  $K$  steps of Langevin updates on  $(\mathbf{x}, \mathbf{z})$  as EMD- $K$ , and we use EMD-1 to describe the DiffInstruct/VSD-equivalent formulation with only one update in  $\mathbf{x}$  as presented in Section 3.4.

## 5.1 ImageNet

We start from showcasing the effect of the key components of EMD, namely noise cancellation, multi-step joint sampling, and reparametrized sampling. We then summarize results on ImageNet  $64 \times 64$  with Karras et al. [47] as teacher, and ImageNet  $128 \times 128$  with Kingma and Gao [29] as teacher.

**Noise cancellation** During our development, we observed the vital importance of canceling the noise after the Langevin update. Even though theoretically speaking our noise cancellation technique does not guarantee reducing the variance of the gradients for learning, we find removing the accumulated noise term from the samples (including the initial diffusion noise  $\epsilon$ ) does give us seemingly clean images empirically. See Fig. 2 for a comparison. These updated  $\mathbf{x}^K$  can be seen as regression targets in Eq. (11). Intuitively speaking, regressing a generator towards clean images should result in more stable training than towards noisy images. Reflected in the training process, canceling the noise significantly decreases the variance in the gradient (Fig. 3a) and boosts the speed of convergence (Fig. 3b). We also compare with another setting where only the noise in the last step gets canceled, which is only marginally helpful.

**Multi-step joint sampling** We scrutinize the effect of multi-step joint update on  $(\epsilon, \mathbf{z})$ . Empirically, we find a constant step size of Langevin dynamics across all noise levels in the  $(\epsilon, \mathbf{z})$ -space works well:  $\gamma = (\gamma_\epsilon, \gamma_z) = (0.4^2, 0.004^2)$ , which simplifies the process of step size tuning. Fig. 1 shows results of running this  $(\epsilon, \mathbf{z})$ -corrector for 300 steps. We can see that the  $(\epsilon, \mathbf{z})$ -corrector removes visual artifacts and improves structure. Fig. 3cd illustrates the relation between the distilled generator’s performance and the number of Langevin steps per distillation iteration, measured by FID and Recall respectively. Both metrics show clear improvement monotonically as the number of Langevin steps increases. Recall is designed for measuring mode coverage [63], and has been widely adopted in the GAN literature. A larger number of Langevin steps encourages better mode coverage, likely because it approximates the mode-covering *forward* KL better. Sampling  $\mathbf{z}$  is more expensive than sampling  $\epsilon$ , requiring back-propagation through the generator  $g_\theta$ . An alternative is to only sample  $\epsilon$  while keeping  $\mathbf{z}$  fixed, with the hope that if  $\mathbf{x}$  does not change dramatically with a finite number of MCMC updates, the initial  $\mathbf{z}$  remains a good approximation of samples from  $\rho_\theta(\mathbf{z}|\mathbf{x})$ . As shown in Fig. 3cd, sampling  $\epsilon$  performs similarly to the joint sampling of  $(\epsilon, \mathbf{z})$  when the number of sampling steps is small, but starts to fall behind with more sampling steps.

**Reparametrized sampling** As shown in Appendix B, the noise cancellation technique does not depend on the reparametrization. One can start from either the score functions of  $(\mathbf{x}, \mathbf{z})$  in Eq. (7) or the score functions of  $(\epsilon, \mathbf{z})$  in Eq. (9) to derive something similar. We conduct a comparison between the two parameterizations for joint sampling,  $(\mathbf{x}, \mathbf{z})$ -corrector and  $(\epsilon, \mathbf{z})$ -corrector.

For the  $(\mathbf{x}, \mathbf{z})$ -corrector, we set the step size of  $\mathbf{x}$  as  $\sigma_t^2 \gamma_\epsilon$  to align the magnitude of update with the one of the  $(\epsilon, \mathbf{z})$ -corrector, and keep

Table 1: EMD-8 on ImageNet  $64 \times 64$ , 100k steps of training

|   | FID ( $\downarrow$ ) | IS ( $\uparrow$ ) |
|---|----------------------|-------------------|
| $(\mathbf{x}, \cdot)/(\epsilon, \cdot)$ | 2.829                | 62.31             |
| $(\mathbf{x}, \mathbf{z})$              | 3.11                 | 61.08             |
| $(\epsilon, \mathbf{z})$                | 2.77                 | 62.98             |



Table 2: Class-conditional genreation on ImageNet  $64 \times 64$ .

| Method                | NFE ( $\downarrow$ ) | FID ( $\downarrow$ ) | Rec. ( $\uparrow$ ) |
|-----------------------|----------------------|----------------------|---------------------|
| <i>Multiple Steps</i> |                      |                      |                     |
| DDIM [12]             | 50                   | 13.7                 |                     |
| EDM-Heun [47]         | 10                   | 17.25                |                     |
| DPM Solver [44]       | 10                   | 7.93                 |                     |
| PD [13]               | 2                    | 8.95                 | 0.65                |
| CD [15]               | 2                    | 4.70                 | 0.64                |
| Multistep CD [18]     | 2                    | 2.0                  | -                   |
| <i>Single Step</i>    |                      |                      |                     |
| BigGAN-deep [65]      | 1                    | 4.06                 | 0.48                |
| EDM [47]              | 1                    | 154.78               | -                   |
| PD [13]               | 1                    | 15.39                | 0.62                |
| BOOT [16]             | 1                    | 16.30                | 0.36                |
| DFNO [17]             | 1                    | 7.83                 | -                   |
| TRACT [14]            | 1                    | 7.43                 | -                   |
| CD-LPIPS [15]         | 1                    | 6.20                 | 0.63                |
| Diff-Instruct [22]    | 1                    | 5.57                 | -                   |
| DMD [23]              | 1                    | 2.62                 | -                   |
| EMD-1 (baseline)      | 1                    | 3.1                  | 0.55                |
| <b>EMD-16 (ours)</b>  | 1                    | <b>2.20</b>          | 0.59                |
| Teacher               | 256                  | 1.43                 | -                   |

Table 3: Class-conditional generation on ImageNet  $128 \times 128$ .

| Method                | NFE ( $\downarrow$ ) | FID ( $\downarrow$ ) | IS ( $\uparrow$ )                |
|-----------------------|----------------------|----------------------|----------------------------------|
| <i>Multiple Steps</i> |                      |                      |                                  |
| Multistep CD [18]     | 8                    | 2.1                  | 164                              |
| Multistep CD [18]     | 4                    | 2.3                  | 157                              |
| Multistep CD [18]     | 2                    | 3.1                  | 147                              |
| <i>Single Step</i>    |                      |                      |                                  |
| CD [15]               | 1                    | 7.0                  | -                                |
| EMD-1 (baseline)      | 1                    | 6.3                  | $134 \pm 2.75$                   |
| <b>EMD-16 (ours)</b>  | 1                    | <b>6.0</b>           | <b><math>140 \pm 2.83</math></b> |
| Teacher               | 512                  | 1.75                 | $171.1 \pm 2.7$                  |

Table 4: FID-30k for text-to-image generation in MSCOCO.  $\dagger$  Results are evaluated by Yin et al. [23].

| Family        | Method                            | Latency ( $\downarrow$ ) | FID ( $\downarrow$ ) |
|---------------|-----------------------------------|--------------------------|----------------------|
| Unaccelerated | DALL-E [66]                       | -                        | 27.5                 |
|               | DALL-E 2 [4]                      | -                        | 10.39                |
|               | Parti-3B [67]                     | 6.4s                     | 8.10                 |
|               | Make-A-Scene [68]                 | 25.0s                    | 11.84                |
|               | GLIDE [69]                        | 15.0s                    | 12.24                |
|               | Imagen [5]                        | 9.1s                     | 7.27                 |
|               | eDiff-1 [70]                      | 32.0s                    | 6.95                 |
| GANs          | StyleGAN-T [71]                   | 0.10s                    | 13.90                |
|               | GigaGAN [72]                      | 0.13s                    | 9.09                 |
| Accelerated   | DPM++ (4 step) $^\dagger$ [45]    | 0.26s                    | 22.36                |
|               | UniPC (4 step) $^\dagger$ [73]    | 0.26s                    | 19.57                |
|               | LCM-LoRA (1 step) $^\dagger$ [74] | 0.09s                    | 77.90                |
|               | LCM-LoRA (4 step) $^\dagger$ [74] | 0.19s                    | 23.62                |
|               | InstaFlow-0.9B $^\dagger$ [55]    | 0.09s                    | 13.10                |
|               | UFOGen [20]                       | 0.09s                    | 12.78                |
|               | DMD (tCFG=3) $^\dagger$ [23]      | 0.09s                    | 11.49                |
|               | EMD-1 (baseline, tCFG=3)          | 0.09s                    | 10.96                |
|               | EMD-1 (baseline, tCFG=2)          | 0.09s                    | 9.78                 |
|               | <b>EMD-8 (ours, tCFG=2)</b>       | 0.09s                    | <b>9.66</b>          |
| Teacher       | SDv1.5 $^\dagger$ [6]             | 2.59s                    | 8.78                 |

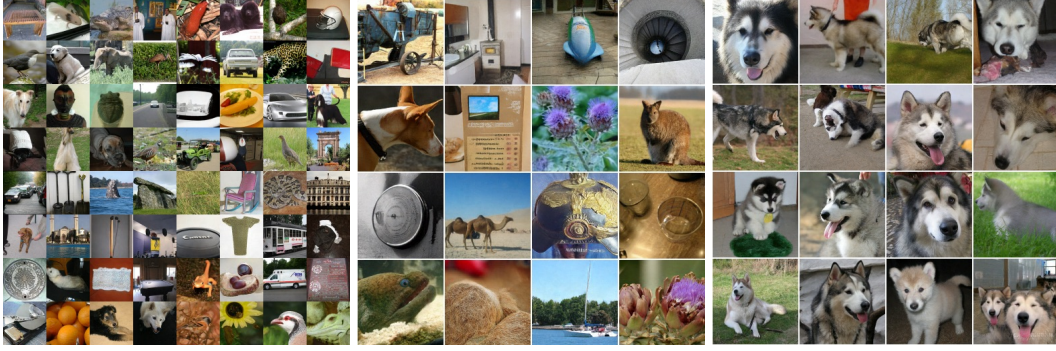
Table 5: CLIP Score in high CFG regime.

| Family      | Method                            | Latency ( $\downarrow$ ) | CLIP ( $\uparrow$ ) |
|-------------|-----------------------------------|--------------------------|---------------------|
| Accelerated | DPM++ (4 step) [45] $^\dagger$    | 0.26s                    | 0.309               |
|             | UniPC (4 step) $^\dagger$ [73]    | 0.26s                    | 0.308               |
|             | LCM-LoRA (1 step) $^\dagger$ [74] | 0.09s                    | 0.238               |
|             | LCM-LoRA (4 step) $^\dagger$ [74] | 0.19s                    | 0.297               |
|             | DMD $^\dagger$ [23]               | 0.09s                    | 0.320               |
|             | <b>EMD-8 (ours)</b>               | 0.09s                    | 0.316               |
| Teacher     | SDv1.5 $^\dagger$ [6]             | 2.59s                    | 0.322               |

the step size of  $\mathbf{z}$  the same (see Appendix B for details). This also promotes numerical stability in  $(\mathbf{x}, \mathbf{z})$ -corrector by canceling the  $\sigma_t^2$  in the denominator of the term  $\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}|\mathbf{z}) = -\frac{\mathbf{x} - \alpha g_{\theta}(\mathbf{z})}{\sigma_t^2}$  in the score function (Eq. (7)). A similar design choice was proposed in Song and Ermon [27]. Also note that adjusting the step sizes in this way results in an equivalence between  $(\epsilon, \cdot)$ -corrector and  $(\mathbf{x}, \cdot)$ -corrector without sampling in  $\mathbf{z}$ , which serves as the baseline for the joint sampling.

Table 1 reports the quantitative comparisons with EMD-8 on ImageNet  $64 \times 64$  after 100k steps of training. While joint sampling with  $(\epsilon, \mathbf{z})$ -corrector improves over  $(\epsilon, \cdot)$ -corrector,  $(\mathbf{x}, \mathbf{z})$ -corrector struggles to even match the baseline. Possible explanations include that the space of  $(\epsilon, \mathbf{z})$  is more MCMC friendly, or it requires more effort on searching for the optimal step size of  $\mathbf{z}$  for the  $(\mathbf{x}, \mathbf{z})$ -corrector. We leave further investigation to future work.

**Comparsion with existing methods** We report the results from our full-fledged method, EMD-16, which utilizes a  $(\epsilon, \mathbf{z})$ -corrector with 16 steps of Langevin updates, and compare with existing approaches. We train for 300k steps on ImageNet  $64 \times 64$ , and 200k steps on ImageNet  $128 \times 128$ . Other hyperparameters can be found in Appendix D. Samples from the distilled generator can be found in Fig. 4. We also include additional samples in Appendix E.1. We summarize the comparison with existing methods for few-step diffusion generation in Table 2 and Table 3 for ImageNet  $64 \times 64$  and ImageNet  $128 \times 128$  respectively. Note that we also tune the baseline EMD-1, which in formulation is equivalent to Diff-Instruct [22], to perform better than their reported numbers. The improvement mainly comes from a fine-grained tuning of learning rates and enabling dropout for both the teacher and student score functions. Our final models outperform existing approaches for one-step distillation of diffusion models in terms of FID scores on both tasks. On ImageNet  $64 \times 64$ , EMD achieves a competitive recall among distribution matching approaches but falls behind trajectory distillation approaches which maintain individual trajectory mappings from the teacher.



(a) ImageNet  $64 \times 64$  multi-class (b) ImageNet  $128 \times 128$  multi-class (c) ImageNet  $128 \times 128$  single-class

Figure 4: ImageNet samples from the distilled 1-step generator. Models are trained class-conditionally with all classes. We provide single-class samples in (c) to demonstrate good mode coverage.



Figure 5: Text-to-image samples from the 1-step student model distilled from Stable Diffusion v1.5.

## 5.2 Text-to-image generation

We further test the potential of EMD on text-to-image models at scale by distilling the Stable Diffusion v1.5 [6] model. Note that the training is image-free and we only use text prompts from the LAION-Aesthetics-6.25+ dataset [75]. On this task, DMD [23] is a strong baseline, which introduced an additional regression loss to VSD or Diff-Instruct to avoid mode collapse. However, we find the baseline without regression loss, or equivalently EMD-1, can be improved by simply tuning the hyperparameter  $t^*$ . Empirically, we find it is better to set  $t^*$  to intermediate noise levels, consistent with the observation from Luo et al. [22]. In Appendix D.4 we discuss the selection of  $t^*$ . The intuition is that by choosing the value of  $t^*$ , we choose a specific denoiser at that noise level for initialization. Other hyperparameters can be found in Appendix D.3.

We evaluate the distilled one-step generator for text-to-image generation with zero-shot generalization on MSCOCO [76] and report the FID-30k in Table 4 and CLIP Score in Table 5. Yin et al. [23] uses the guidance scale of 3.0 to compose the classifier-free guided teacher score (we refer to this guidance scale of teacher as tCFG) in the learning gradient of DMD, for it achieves the best FID for DDIM sampler. However, we find EMD achieves a lower FID at the tCFG of 2.0. Our method, EMD-8, trained on 256 TPU-v5e for 5 hours (5000 steps), achieves the FID=9.66 for one-step text-to-image generation. Using a higher tCFG, similar to DMD, produces a model with competitive CLIP Score. In Fig. 5, we include some samples for qualitative evaluation. Additional qualitative results (Tables 14 and 15), as well as side-by-side comparisons (Tables 10 to 13) with trajectory-based distillation baselines [55, 74] and adversarial distillation baselines [21] can be found in Appendix E.2.

Table 6: Training steps per second in ablations for computation overhead in ImageNet  $64 \times 64$ 

| Algorithmic Ablation   | sec/step |
|--|----------|
| Student score matching only  | 0.303    |
| Generator update for EMD-1 based on $(\epsilon, \mathbf{z})$ -corrector                          | 0.303    |
| Generator update for EMD-2 based on $(\epsilon, \mathbf{z})$ -corrector                          | 0.417    |
| Generator update for EMD-4 based on $(\epsilon, \mathbf{z})$ -corrector                          | 0.556    |
| Generator update for EMD-8 based on $(\epsilon, \mathbf{z})$ -corrector                          | 0.714    |
| Generator update for EMD-16 based on $(\epsilon, \mathbf{z})$ -corrector                         | 1.111    |
| EMD-16 ( student score matching + generator update based on $(\epsilon, \mathbf{z})$ -corrector) | 1.515    |
| Baseline Diff-Instruct (student score matching + generator update)                               | 0.703    |

### 5.3 Computation overhead in training

Despite EMD being more expensive per training iteration compared to the baseline approach Diff-Instruct, we find the performance gain of EMD cannot be realized by simply running Diff-Instruct for the same amount of time or even longer than EMD. In fact, the additional computational cost that EMD introduced is moderate even with the most expensive EMD-16 setting. In Table 6 we report some quantitative measurement of the computation overhead. Since it is challenging to time each python method’s wall-clock time in our infrastructure, we instead logged the sec/step for experiments with various algorithmic ablations on ImageNet  $64 \times 64$ . EMD-16 only doubles the wall-clock time of Diff-Instruct when taking all other overheads into account.

## 6 Discussion and limitation

We present EMD, a maximum likelihood-based method that leverages EM framework with novel sampling and optimization techniques to learn a one-step student model whose marginal distributions match the marginals of a pretrained diffusion model. EMD demonstrates strong performance in class-conditional generation on ImageNet and text-to-image generation. Despite exhibiting compelling results, EMD has a few limitations that call for future work. Empirically, we find that EMD still requires the student model to be initialized from the teacher model to perform competitively, and is sensitive to the choice of  $t^*$  (fixed timestep conditioning that repurposes the diffusion denoiser to become a one-step genertor) at initialization. While training a student model entirely from scratch is supported theoretically by our framework, empirically we were unable to achieve competitive results. Improving methods to enable generation from randomly initialized generator networks with distinct architectures and lower-dimensional latent variables is an exciting direction of future work. Although being efficient in inference, EMD introduces additional computational cost in training by running multiple sampling steps per iteration, and the step size of MCMC sampling can require careful tuning. There remains a fundamental trade-off between training cost and model performance. Analysis and further improving on the Pareto frontier of this trade-off would be interesting for future work.

## Acknowledgement

We thank Jonathan Heek and Lucas Theis for their valuable discussion and feedback. We also thank Tianwei Yin for helpful sharing of experimental details in his work. Sirui would like to thank Tao Zhu, Jiahui Yu, and Zhishuai Zhang for their support in a prior internship at Google DeepMind.

## References

- [1] 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.
- [2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

- [3] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.
- [8] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL <https://openai.com/research/video-generation-models-as-world-simulators>.
- [9] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.
- [10] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.
- [11] Mengjiao Yang, KwangHwan Cho, Amil Merchant, Pieter Abbeel, Dale Schuurmans, Igor Mordatch, and Ekin Dogus Cubuk. Scalable diffusion for materials generation. *arXiv preprint arXiv:2311.09235*, 2023.
- [12] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [13] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.
- [14] 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.
- [15] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023.
- [16] 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.
- [17] 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.
- [18] Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.
- [19] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021.



- [20] Yanwu Xu, Yang Zhao, Zhisheng Xiao, and Tingbo Hou. Ufogen: You forward once large scale text-to-image generation via diffusion gans. *arXiv preprint arXiv:2311.09257*, 2023.
- [21] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.
- [22] 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, 2023.
- [23] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.
- [24] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39 (1):1–22, 1977.
- [25] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.
- [26] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Pro-lificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.
- [27] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- [28] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.
- [29] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36, 2023.
- [30] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.
- [31] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.
- [32] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.
- [33] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644. PMLR, 2016.
- [34] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [35] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9155–9164, 2018.
- [36] Erik Nijkamp, Ruiqi Gao, Pavel Sountsov, Srinivas Vasudevan, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Mcmc should mix: Learning energy-based model with neural transport latent space mcmc. In *International Conference on Learning Representations*, 2021.
- [37] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5272–5280, 2020.
- [38] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.



- [39] Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2: 1122–1128, 2006.
- [40] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- [41] Thuan Hoang Nguyen and Anh Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. *arXiv preprint arXiv:2312.05239*, 2023.
- [42] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2020.
- [43] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.
- [44] 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.
- [45] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.
- [46] 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.
- [47] 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.
- [48] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [49] 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.
- [50] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *arXiv preprint arXiv:2306.00980*, 2023.
- [51] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.
- [52] 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.
- [53] Jianbin Zheng, Minghui Hu, Zhongyi Fan, Chaoyue Wang, Changxing Ding, Dacheng Tao, and Tat-Jen Cham. Trajectory consistency distillation. *arXiv preprint arXiv:2402.19159*, 2024.
- [54] 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.
- [55] 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.

- [56] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*, 2024.
- [57] Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.
- [58] 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.
- [59] 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.
- [60] 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.
- [61] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
- [62] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.
- [63] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.
- [64] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [65] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.
- [66] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.
- [67] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.
- [68] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision*, pages 89–106. Springer, 2022.
- [69] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.
- [70] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.
- [71] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. arxiv e-prints. *arXiv preprint arXiv:1812.04948*, 2018.

- [72] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10124–10134, 2023.
- [73] 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, 2024.
- [74] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023.
- [75] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.
- [76] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
- [77] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- [78] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023.

## A Expectation-Maximization

To learn a latent variable model  $p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})$ ,  $p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{x}, \mathbf{z})d\mathbf{z}$  from a target distribution  $q(\mathbf{x})$ , the EM-like transformation on the gradient of the log-likelihood function is:

$$\begin{aligned}
\nabla_{\theta}\mathcal{L}(\theta) &= \nabla_{\theta}\mathbb{E}_{q(\mathbf{x})}[\log p_{\theta}(\mathbf{x})] \\
&= \mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})}[\mathbb{E}_{q(\mathbf{x})}[\nabla_{\theta}\log p_{\theta}(\mathbf{x})]] \\
&= \mathbb{E}_{q(\mathbf{x})p_{\theta}(\mathbf{z}|\mathbf{x})}[\nabla_{\theta}\log p_{\theta}(\mathbf{x}) + \nabla_{\theta}\log p_{\theta}(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q(\mathbf{x})p_{\theta}(\mathbf{z}|\mathbf{x})}[\nabla_{\theta}\log p_{\theta}(\mathbf{x}, \mathbf{z})].
\end{aligned} \tag{15}$$

Line 3 is due to the simple equality that  $\mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})}[\nabla_{\theta}\log p_{\theta}(\mathbf{z}|\mathbf{x})] = 0$ .

## B Reparametrized sampling and noise cancellation

**Reparametrization.** The EM-like algorithm we propose requires joint sampling of  $(\mathbf{x}, \mathbf{z})$  from  $p_{\theta}(\mathbf{x}, \mathbf{z})$ . Similar to [36, 42], we utilize a reparameterization of  $\mathbf{x}$  and  $\mathbf{z}$  to overcome challenges in joint MCMC sampling, such as slow convergence and complex step size tuning. Notice that  $\mathbf{x} = \alpha g_{\theta}(\mathbf{z}) + \sigma \epsilon$  defines a deterministic mapping from  $(\epsilon, \mathbf{z})$  to  $(\mathbf{x}, \mathbf{z})$ . Then by *change of variable* we have:

$$\begin{aligned}
\rho_{\theta}(\epsilon, \mathbf{z})d\epsilon d\mathbf{z} &= \rho_{\theta}(\mathbf{x}, \mathbf{z})d\mathbf{x}d\mathbf{z} \\
&= \frac{q(\mathbf{x})}{p_{\theta}(\mathbf{x})}p_{\theta}(\mathbf{x}, \mathbf{z})d\mathbf{x}d\mathbf{z} \\
&= \frac{q(\alpha g_{\theta}(\mathbf{z}) + \sigma \epsilon)}{p_{\theta}(\alpha g_{\theta}(\mathbf{z}) + \sigma \epsilon)}p_{\theta}(\epsilon, \mathbf{z})d\epsilon d\mathbf{z} \\
\Rightarrow \rho_{\theta}(\epsilon, \mathbf{z}) &= \frac{q(\alpha g_{\theta}(\mathbf{z}) + \sigma \epsilon)}{p_{\theta}(\alpha g_{\theta}(\mathbf{z}) + \sigma \epsilon)}p(\epsilon)p(\mathbf{z}),
\end{aligned} \tag{16}$$

where  $p(\epsilon)$  and  $p(\mathbf{z})$  are standard Normal distributions.

The score functions become

$$\begin{aligned}\nabla_{\epsilon} \log \rho_{\theta}(\epsilon, \mathbf{z}) &= \sigma(\nabla_{\mathbf{x}} \log q(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})) - \epsilon, \\ \nabla_{\mathbf{z}} \log \rho_{\theta}(\epsilon, \mathbf{z}) &= \alpha(\nabla_{\mathbf{x}} \log q(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})) \nabla_{\mathbf{z}} g_{\theta}(\mathbf{z}) - \mathbf{z}.\end{aligned}\quad (17)$$

**Noise cancellation.** The single-step Langevin update on  $\epsilon$  is then:

$$\epsilon^{i+1} = (1 - \gamma)\epsilon^i + \gamma\sigma\nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^i)}{p_{\theta}(\mathbf{x}^i)} + \sqrt{2\gamma}\mathbf{n}^i. \quad (18)$$

Interestingly, we find the particular form of  $\nabla_{\epsilon} \log \rho(\epsilon, \mathbf{z})$  results in a closed-form accumulation of multi-step updates

$$\epsilon^{i+1} = (1 - \gamma)^{i+1}\epsilon^0 + \gamma \sum_{k=0}^i (1 - \gamma)^{i-k} \sigma \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^i)}{p_{\theta}(\mathbf{x}^i)} + \sum_{k=0}^i (1 - \gamma)^{i-k} \sqrt{2\gamma}\mathbf{n}^k, \quad (19)$$

which, after the push-forward, gives us

$$\begin{aligned}\mathbf{x}^{i+1} &= \alpha g(\mathbf{z}^{i+1}) + \underbrace{\gamma \sum_{k=0}^i (1 - \gamma)^{i-k} \sigma^2 \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^i)}{p_{\theta}(\mathbf{x}^i)}}_{\text{drift}} \\ &\quad + \underbrace{(1 - \gamma)^{i+1} \sigma \epsilon^0 + \sum_{k=0}^i (1 - \gamma)^{i-k} \sqrt{2\gamma} \sigma \mathbf{n}^k}_{\text{noise}},\end{aligned}\quad (20)$$

$$\mathbf{x}^1 = \alpha g(\mathbf{z}^1) + \underbrace{\gamma \sigma^2 \nabla_{\mathbf{x}} \log \frac{q(\mathbf{x}^0)}{p_{\theta}(\mathbf{x}^0)}}_{\text{drift}} + \underbrace{(1 - \gamma) \sigma \epsilon^0 + \sqrt{2\gamma} \sigma \mathbf{n}^0}_{\text{noise}} \quad (21)$$

As  $\mathbf{x}^{i+1}$  is effectively a regression target in Eq. (6), and the expected value of the *noise* is  $\mathbf{0}$ , we can remove it without biasing the gradient. Empirically, we find book-keeping the sampled noises in the MCMC chain and canceling these noises after the loop significantly stabilize the training of the generator network.

The same applies to the  $(\mathbf{x}, \mathbf{z})$  sampling (with step size  $\gamma\sigma^2$ ):

$$\begin{aligned}\mathbf{x}^{i+1} &= \mathbf{x}^i + \gamma\sigma^2(\nabla_{\mathbf{x}} \log q(\mathbf{x}^i) - \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}^i)) - \gamma(\mathbf{x}^i - \alpha g(\mathbf{z}^i)) + \sqrt{2\gamma}\mathbf{n}^i \\ &= \gamma \sum_{k=1}^i (1 - \gamma)^{i-k} \alpha g(\mathbf{z}^k) + \gamma \sum_{k=0}^i (1 - \gamma)^{i-k} \sigma^2 (\nabla_{\mathbf{x}} \log q(\mathbf{x}^i) - \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}^i)) \\ &\quad + (1 - \gamma)^i \alpha g(\mathbf{z}^0) + \underbrace{(1 - \gamma)^{i+1} \sigma \epsilon^0 + \sum_{k=0}^i (1 - \gamma)^{i-k} \sqrt{2\gamma} \sigma \mathbf{n}^k}_{\text{noises}}.\end{aligned}\quad (22)$$

## C JAX-style Code

We provide JAX-style code to illustrate how the MCMC correction and generator loss work.

**MCMC correction** The MCMC corrector is mainly a loop function that iterates with Langevin updates on  $\epsilon$  and  $\mathbf{z}$ . The noises for the Langevin updates on  $\epsilon$  are pre-sampled before the loop body. They are then integrated by weights and subtracted from the MCMC corrected  $\hat{\epsilon}$  for noise cancellation. Finally, the corrected and noise-canceled  $\hat{\epsilon}$  is used to parameterize  $\hat{\mathbf{x}}$ .

```
1 def eps_corrector(self, eps, z, lamdb, rng):
2     ### collecting parameters ###
3     ld_steps = self.config.model.ld_steps
4     z_step_size = self.config.model.ld_size_z
5     eps_step_size = self.config.model.ld_size_eps
6     sigma = jnp.sqrt(nn.sigmoid(lamdb))
7     sigma = utils.broadcast_from_left(sigma, eps.shape)
```

```

8     alpha = jnp.sqrt(nn.sigmoid(-lambd))
9     alpha = utils.broadcast_from_left(alpha, eps.shape)
10    z_base_rng, eps_base_rng = jax.random.split(rng)
11
12    ### pre-sampling eps noises ###
13    eps_noises = jax.random.normal(eps_base_rng, (ld_steps,) + eps.
14    shape)
15
16    def loop_body(step, val):
17        ### calculating scores ###
18        eps, z = val
19        x, _ = self.sample_g(z, None)
20        xt = alpha * x + sigma * eps
21        teacher_model_output = self._run_model(
22            xt=xt,
23            lambd=lambd,
24            model_fn=self.model_fns['teacher'],
25        )
26        s_model_output = self._run_model(
27            xt=xt, lambd=lambd, model_fn=self.model_fns['s'],
28        )
29        teacher_eps = teacher_model_output['model_eps']
30        s_eps = s_model_output['model_eps']
31        diff = - jax.lax.stop_gradient(teacher_eps - s_eps) / sigma
32        grad_z = self.grad_z_g(z, diff, alpha) # alpha * diff * grad_z(g
33        (z))
34        z_score = grad_z - z
35        eps_score = diff * sigma - eps
36
37        ### Langevin update on z ###
38        z_rng = jax.random.fold_in(z_base_rng, step)
39        z_noise = jax.random.normal(z_rng, z.shape)
40        z_mean = z + z_step_size**2 * z_score
41        z_next = z_mean + z_noise * jnp.sqrt(2) * z_step_size
42        z_next = jax.lax.stop_gradient(z_next)
43
44        ### Langevin update on eps ###
45        eps_noise = eps_noises[step]
46        eps_mean = eps + eps_score * eps_step_size**2
47        eps_next = eps_mean + eps_noise * jnp.sqrt(2) * eps_step_size
48        eps_next = jax.lax.stop_gradient(eps_next)
49
50        return eps_next, z_next
51
52    ### ld_steps Langevin updates ###
53    eps_hat, z_hat = jax.lax.fori_loop(0, ld_steps, loop_body, (eps, z
54    ))
55    ### preparing noise weights ###
56    step_weights = (1 - eps_step_size ** 2) ** jnp.arange(ld_steps)
57    [::-1]
58    step_weights = utils.broadcast_from_left(step_weights, eps_noises.
59    shape)
60    ### noise cancellation ###
61    eps_hat = eps_hat - (1 - eps_step_size ** 2) ** ld_steps * eps
62    eps_hat = eps_hat - jnp.sqrt(2) * jnp.sum(eps_step_size *
63    step_weights * eps_noises, axis=0)
64    ### parametrizing x_hat ###
65    x_h_hat, _ = self.sample_g(z_hat, None)
66    x_hat = x_h_hat + sigma * eps_hat / alpha
67    x_hat = jax.lax.stop_gradient(x_hat)
68    return x_hat, z_hat

```

Listing 1: MCMC corrector for  $(\epsilon, z)$



**Generator loss** The generator loss is simply an MSE loss between  $\hat{\mathbf{x}}$  and  $g_{\theta}(\hat{\mathbf{z}})$

```

1 def x_loss_g(self, eps, z, lambd, rng):
2     ### MCMC correction ###
3     x_hat, z_hat = self.eps_corrector(eps, z, lambd, rng)
4     x_z_hat, _ =
5     ### forwarding generator network ###
6     self.sample_g(z_hat, None)
7     ### calculating loss ###
8     loss = jnp.square(jax.lax.stop_gradient(x_hat) - x_z_hat)
9     loss = utils.meanflat(loss)
10    return loss, x_hat, x_z_hat

```

Listing 2: Generator loss

## D Implementation details

### D.1 ImageNet 64×64

We train the teacher model using the best setting of EDM [47] with the ADM UNet architecture [77]. We inherit the noise schedule and the score matching weighting function from the teacher. We run the distillation training for 300k steps (roughly 8 days) on 64 TPU-v4. We use  $(\epsilon, \mathbf{z})$ -corrector, in which both the teacher and the student score networks have a dropout probability of 0.1. We list other hyperparameters in Table 7. Instead of listing  $t^*$ , we list the corresponding log signal-to-noise ratio  $\lambda^*$ .

Table 7: Hyperparameters for EMD on ImageNet 64×64.

| $lr_g$             | $lr_s$             | batch size | Adam $b_1$ | Adam $b_2$ | $\gamma_{\epsilon}$ | $\gamma_{\mathbf{z}}$ | $K$ | $\lambda^*$ | $\tilde{w}(t)$                  |
|--------------------|--------------------|------------|------------|------------|---------------------|-----------------------|-----|-------------|---------------------------------|
| $2 \times 10^{-6}$ | $1 \times 10^{-5}$ | 128        | 0.0        | 0.99       | $0.4^2$             | $0.004^2$             | 16  | -3.2189     | $\frac{\sigma_t^2}{\alpha_t^2}$ |

### D.2 ImageNet 128×128

We train the teacher model following the best setting of VDM++ [29] with the ‘U-ViT, L’ architecture [78]. We use the ‘cosine-adjusted’ noise schedule [78] and ‘EDM monotonic’ weighting for student score matching. We run the distillation training for 200k steps (roughly 10 days) on 128 TPU-v5p. We use  $(\epsilon, \mathbf{z})$ -corrector, in which both the teacher and the student score networks have a dropout probability of 0.1. We list other hyperparameters in Table 8.

Table 8: Hyperparameters for EMD on ImageNet 128×128.

| $lr_g$             | $lr_s$             | batch size | Adam $b_1$ | Adam $b_2$ | $\gamma_{\epsilon}$ | $\gamma_{\mathbf{z}}$ | $K$ | $\lambda^*$ | $\tilde{w}(t)$                |
|--------------------|--------------------|------------|------------|------------|---------------------|-----------------------|-----|-------------|-------------------------------|
| $2 \times 10^{-6}$ | $1 \times 10^{-5}$ | 1024       | 0.0        | 0.99       | $0.4^2$             | $0.004^2$             | 16  | -6          | $\frac{\sigma_t^2}{\alpha_t}$ |

### D.3 Text-to-image generation

We adopt the public checkpoint of Stable Diffusion v1.5 [6] as the teacher. We inherit the noise schedule from the teacher model. The student score matching uses the same weighting function as the teacher. We list other hyperparameters in Table 9.

Table 9: Hyperparameters for EMD on Text-to-image generation.

| $lr_g$             | $lr_s$             | batch size | Adam $b_1$ | Adam $b_2$ | $\gamma_{\epsilon}$ | $\gamma_{\mathbf{z}}$ | $K$ | $t^*$ | $\tilde{w}(t)$                |
|--------------------|--------------------|------------|------------|------------|---------------------|-----------------------|-----|-------|-------------------------------|
| $2 \times 10^{-6}$ | $1 \times 10^{-5}$ | 1024       | 0.0        | 0.99       | $0.3^2$             | $0.005^2$             | 8   | 500   | $\frac{\sigma_t^2}{\alpha_t}$ |

#### D.4 Choosing $t^*$ and $\lambda^*$

The intuition is that by choosing the value of  $t^*$ , we choose a specific denoiser at that noise level. When parametrizing  $t$ , the log-signal-to-noise ratio  $\lambda$  is more useful when designing noise schedules, a strictly monotonically decreasing function [28]. Due to the monotonicity,  $\lambda^*$  is an alternative representation for  $t^*$  that actually reflects the noise levels more directly.

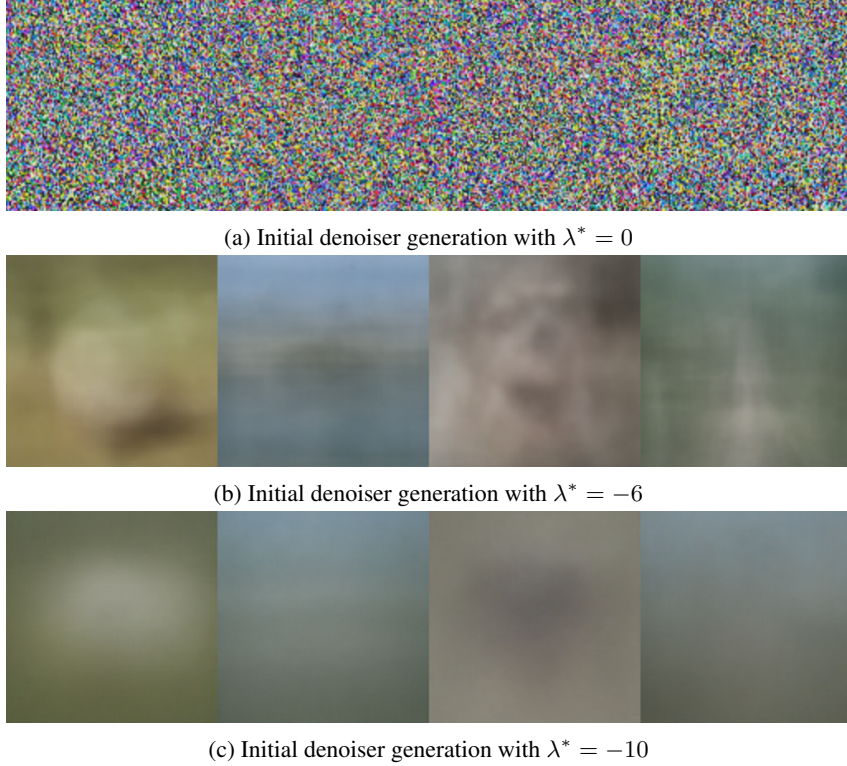


Figure 6: Initial denoiser generation with different  $\lambda^*$ .

Fig. 6 shows the denoiser generation at the 0th training iteration for different  $\lambda^*$  in ImageNet  $128 \times 128$ . When  $\lambda^* = 0$ , the generated images are no different from Gaussian noises. When  $\lambda^* = -6$ , the generated images have more details than  $\lambda^* = -10$ . In the context of EMD, these samples help us understand the initialization of MCMC. According to our experiments, setting  $\lambda^* \in [-6, -3]$  results in similar performance. For the numbers reported in the manuscript, we used the same  $\lambda^*$  as the baseline Diff-Instruct on ImageNet  $64 \times 64$  and only did a very rough grid search on ImageNet  $128 \times 128$  and Text-to-image.

## E Additional qualitative results

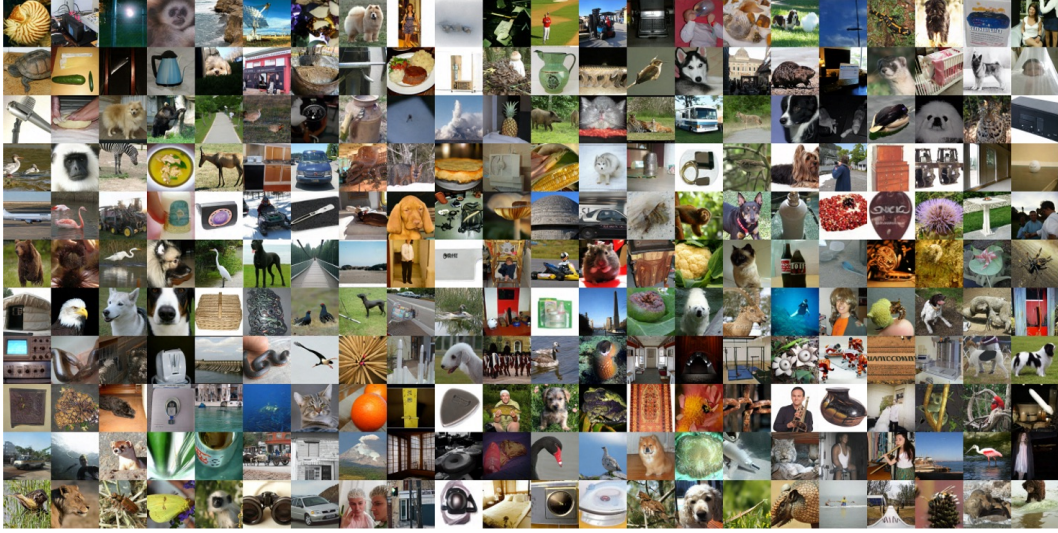
### E.1 Additional ImageNet results

In this section, we present additional qualitative samples for our one-step generator on ImageNet  $64 \times 64$  and ImageNet  $128 \times 128$  in Fig. 7 to help further evaluate the generation quality and diversity.

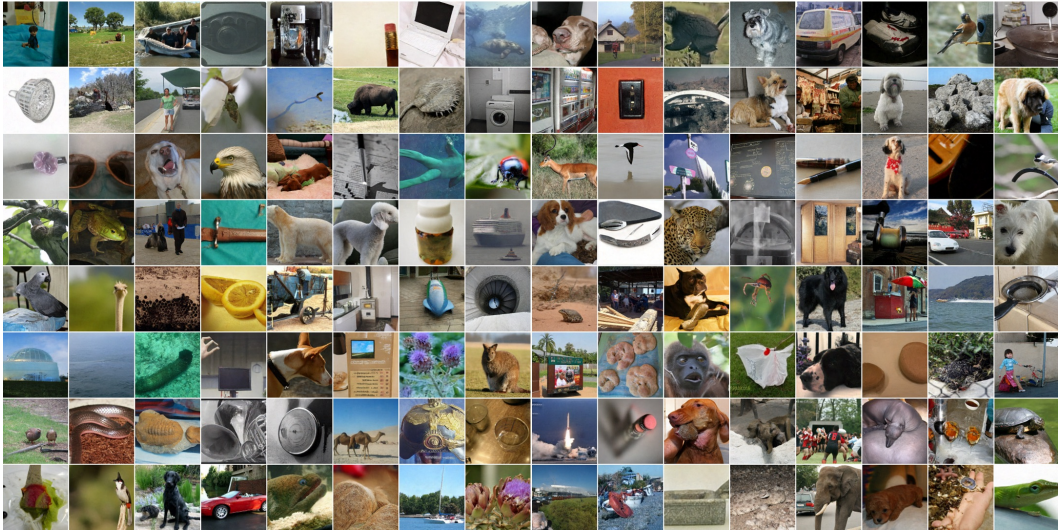
### E.2 Additional text-to-image results

In this section, we present additional qualitative samples from our one-step generator distilled from Stable Diffusion 1.5. In Table 10, 11, 12, and 13, we visually compare the sample quality of our method with open-source competing methods for few- or single-step generation. We also include the

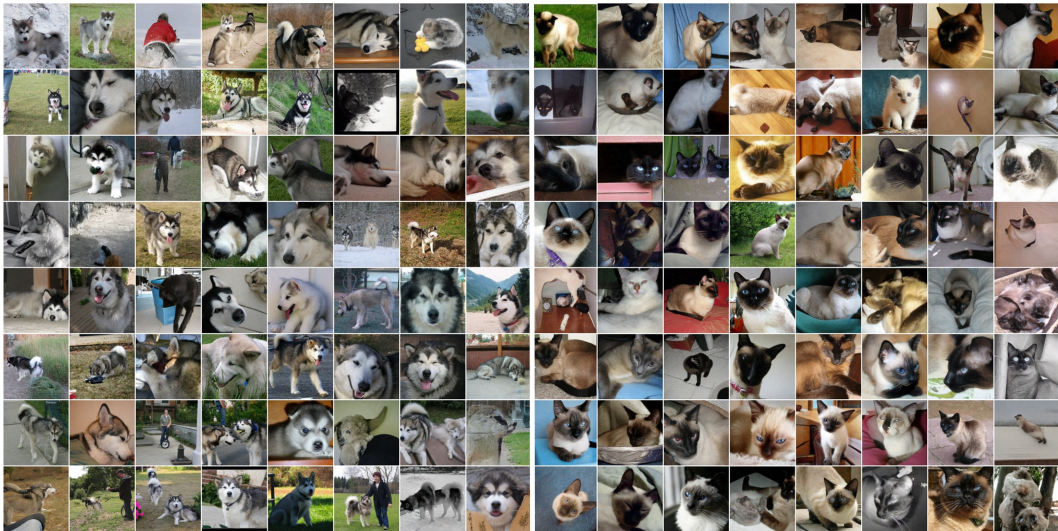




(a) ImageNet  $64 \times 64$  Multi-class



(b) ImageNet  $128 \times 128$  Multi-class



(c) ImageNet  $128 \times 128$  Single-class (Left: Husky, right: Siamese)

Figure 7: Additional qualitative results for ImageNet

teacher model in our comparison. We use the public checkpoints of LCM<sup>1</sup> and InstaFlow<sup>2</sup>, where both checkpoints share the same Stable Diffusion 1.5 as teachers. Note that the SD-turbo results are obtained from the public checkpoint<sup>3</sup> fine-tuned from Stable Diffusion 2.1, which is different from our teacher model.

From the comparison, we observe that our model significantly outperforms distillation-based methods including LCM and InstaFlow, and it demonstrates better diversity and quality than GAN-based SD-turbo. The visual quality is on-par with 50-step generation from the teacher model.

We show additional samples from our model on a more diverse set of prompts in Table 14 and 15.

---

<sup>1</sup><https://huggingface.co/latent-consistency/lcm-lora-sdv1-5>

<sup>2</sup>[https://huggingface.co/XCLiu/instaflow\\_0\\_9B\\_from\\_sd\\_1\\_5](https://huggingface.co/XCLiu/instaflow_0_9B_from_sd_1_5)

<sup>3</sup><https://huggingface.co/stabilityai/sd-turbo>





Teacher (50 step)



EMD (1 step)



LCM (1 step)



LCM (2 steps)



SD-Turbo (1 step)



InstaFlow (1 step)

Table 10: Prompt: *Dog graduation at university.*





Teacher (50 step)



EMD (1 step)



LCM (1 step)



LCM (2 steps)



SD-Turbo (1 step)



InstaFlow (1 step)

Table 11: Prompt: *3D animation cinematic style young caveman kid, in its natural environment.*



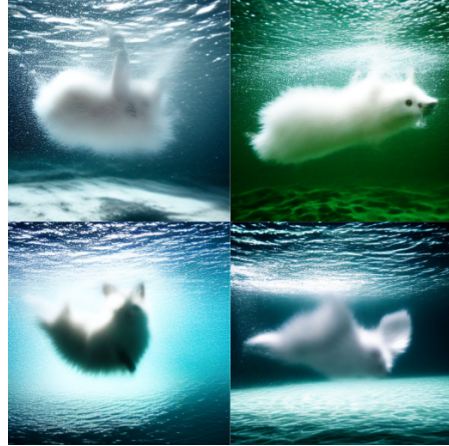
Teacher (50 step)



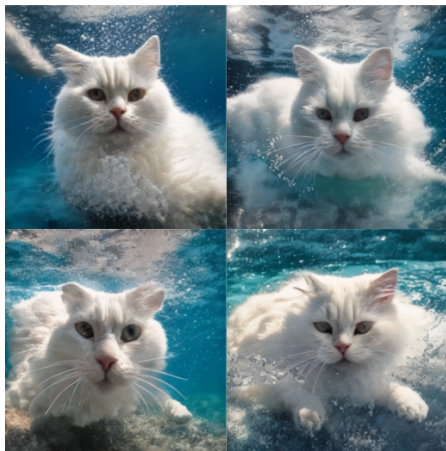
EMD (1 step)



LCM (1 step)



LCM (2 steps)



SD-Turbo (1 step)



InstaFlow (1 step)

Table 12: Prompt: *An underwater photo portrait of a beautiful fluffy white cat, hair floating. In a dynamic swimming pose. The sun rays filters through the water. High-angle shot. Shot on Fujifilm X.*

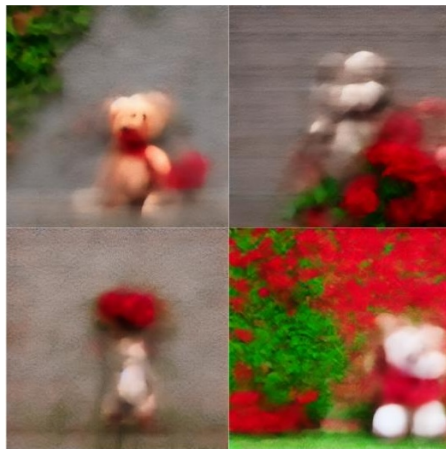




Teacher (50 step)



EMD (1 step)



LCM (1 step)



LCM (2 steps)



SD-Turbo (1 step)



InstaFlow (1 step)

Table 13: Prompt: *A minimalist Teddy bear in front of a wall of red roses.*





*A close-up photo of a intricate beautiful natural landscape of mountains and waterfalls.*



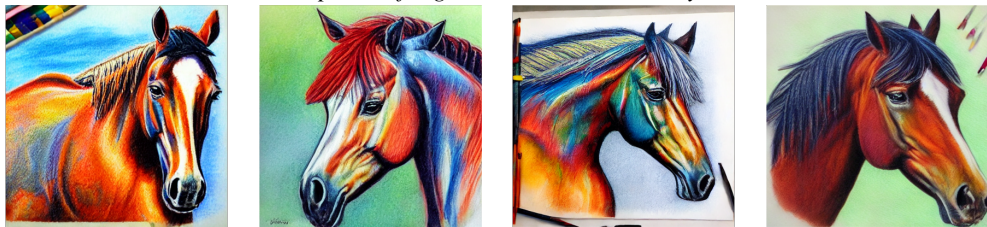
*A hyperrealistic photo of a fox astronaut; perfect face, artstation.*



*Large plate of delicious fried chicken, with a side of dipping sauce, realistic advertising photo, 4k.*



*A DSLR photo of a golden retriever in heavy snow.*



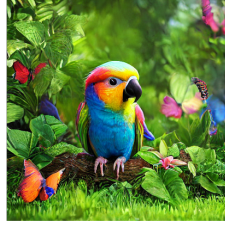
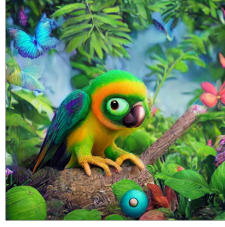
*Masterpiece color pencil drawing of a horse, bright vivid color.*



*Oil painting of a wise old man with a white beard in the enchanted and magical forest.*

Table 14: Additional qualitative results of EMD. Zoom-in for better viewing.





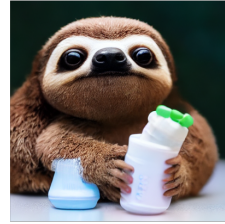
*3D render baby parrot, Chibi, adorable big eyes. In a garden with butterflies, greenery, lush whimsical and soft, magical, octane render, fairy dust.*



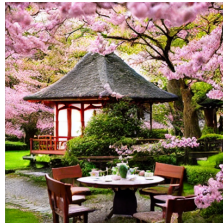
*Dreamy puppy surrounded by floating bubbles.*



*A painting of an adorable rabbit sitting on a colorful splash.*



*Macro photo of a miniature toy sloth drinking a soda, shot on a light pastel cyclorama.*



*A traditional tea house in a tranquil garden with blooming cherry blossom trees.*



*Three cats having dinner at a table at new years eve, cinematic shot, 8k.*

Table 15: Additional qualitative results of EMD. Zoom-in for better viewing.