DIFF-INSTRUCT*: TOWARDS HUMAN-PREFERRED ONE-STEP TEXT-TO-IMAGE GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we introduce the Diff-Instruct* (DI*), an image data-free approach for building one-step text-to-image generative models that align with human preference while maintaining the ability to generate highly realistic images. We frame human preference alignment as online reinforcement learning using human feedback (RLHF), where the goal is to maximize the reward function while regularizing the generator distribution to remain close to a reference diffusion process. Unlike traditional RLHF approaches, which rely on the KL divergence for regularization, we introduce a novel score-based divergence regularization, which leads to significantly better performances. Although the direct calculation of this divergence remains intractable, we demonstrate that we can efficiently compute its *gradient* by deriving an equivalent yet tractable loss function. Remarkably, with Stable Diffusion V1.5 as the reference diffusion model, DI* outperforms *all* previously leading models by a large margin. When using the 2.6B Stable Diffusion XL architecture, the DI* results in a solid human-preferred one-step model that is able to generate aesthetic images of 1024×1024 resolutions. When using the 0.6B PixelArt-α model as the reference diffusion, DI* achieves a new record Aesthetic Score of 6.30 and an Image Reward of 1.31 with only a single generation step, almost doubling the scores of the rest of the models with similar sizes. It also achieves an HPSv2.0 score of 28.70, establishing a new state-of-the-art benchmark, with a better layout, richer details, and aesthetic colors.

1 INTRODUCTIONS

Deep generative models have made substantial progress these years, largely transforming the content creation and editing across various domains (Karras et al., 2020; Nichol & Dhariwal, 2021; Poole et al., 2022; Kim et al., 2022; Tashiro et al., 2021; Meng et al., 2021; Couairon et al., 2022; Ramesh et al., 2022; Esser et al., 2024). These models demonstrated exceptional capabilities in generating high-resolution outputs, such as photorealistic images, videos, audio, and 3D assets (Oord et al., 2016; Ho et al., 2022; Poole et al., 2022; Brooks et al., 2024), and others (Zhang et al., 2023a; Xue et al., 2023; Luo & Zhang, 2024; Luo et al., 2023b; Zhang et al., 2023b; Feng et al., 2023; Deng et al., 2024; Luo et al., 2024d; Geng et al., 2024; Wang et al., 2024; Pokle et al., 2022).

Within this field, two types of generative models have gained significant attention, diffusion models and one-step generators. Diffusion models (DMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020), or score-based generative models (Song et al., 2020), first progressively corrupt data with diffusion processes and then train models to approximate the score functions of the noisy data distributions across varying noise levels. The learned score functions can be used in the reverse process to generate high-quality samples by iterative denoising the noisy samples through stochastic differential equations. While DMs can produce high-quality outputs, they often require a large number of model evaluations, which limits their efficiency in applications.

Different from diffusion models, one-step generators (Zheng & Yang, 2024; Kang et al., 2023a; Sauer et al., 2023a; Yin et al., 2024; Zhou et al., 2024a; Luo et al., 2024c) have emerged as a highly efficient alternative to multi-step diffusion models. Unlike DMs, one-step generative models directly transform latent noises to output samples with a single neural network forward pass. This mechanism significantly reduced the inference cost, making them ideal for real-time applications such as text-to-image and text-to-video generations. Many existing works have demonstrated the

Figure 1: None cherry-picked generated images from one-step 0.6B DiT-DI* model with an record-breaking HPSv2.0 score of 28.70. After being trained with Diff-Instruct*, the images show better layouts, rich colors, vivid details, and aesthetic appearance, making them favored in terms of human preferences. Refer to the Appendix B.1 for the prompts used in comparison.

Figure 2: A visual comparison of our 0.6B DiT-DI* and 0.86B SD1.5-DI* models against other text-t0-image models. Refer to the Appendix B.1 for the prompts used in comparison.

leading performances of one-step text-to-image generators (Zhou et al., 2024a; Yin et al., 2024; Luo et al., 2024c) by employing diffusion distillation (Luo, 2023). However, these works focus on matching the generator distributions with pretrained diffusion models, without considering the critical challenge of teaching one-step text-to-image models to satisfy human preferences, which is one of the most important needs in the age of human-centric AI.

To close this gap, we introduce Diff-Instruct* (DI*), a novel approach to train human-preferred one-step generators. Inspired by the success of reinforcement learning using human feedback (RLHF) in training large language models (Christiano et al., 2017; Ouyang et al., 2022), we frame the human-preference alignment problem as maximizing the rewards with a score-based divergence constraint. This yields generated samples that not only adhere to user prompts but also improve in aesthetic quality. Our approach differs from traditional RLHF methods, which rely on Kullback-Leibler (KL) divergence for distribution regularization. Instead, we propose score-based divergences, which offer more stable training dynamics and better final performance. While the direct computation of the score-based divergence is intractable, we develop a novel solution by deriving a tractable gradient computation method, leading to a tractable pseudo-loss that stands equivalent to the intractable one in the gradient perspective. With DI*, we are able to train large-scale human-preferred one-step text-to-image generative models.

In our evaluation, our best 0.6B DiT-DI*-1step model, using diffusion transformer (DiT) (Peebles & Xie, 2022) as generator architecture and PixelArt-α (Chen et al., 2023) as reference diffusion model, has set a new benchmark on the COCO-2017 validation prompts dataset. It achieves an ImageReward (Xu et al., 2023a) of 1.31 and an Aesthetic Score(Schuhmann, 2022) of 6.30, significantly outperforming the previous leading models such as 2.6B SDXL-DMD2 4-step models (Yin et al., 2024) and Stable Diffusion XL (Podell et al., 2023) with 25 model steps by **50.5%** and **77.0%**, respectively. Moreover, this model also reaches a new high with Human Preference Score V2.0 (Wu et al., 2023) of **28.70**, surpassing leading models like SDXL-DMD2, the PixelArt-α, DALL-E 2 (Ramesh et al., 2022) and the Stable Diffusion XL with Refiners. When using Stable Diffusion V1.5 as the reference model and the same UNet architecture for the generator, our SD1.5-DI*-1step model with only 0.86B parameters exceeds the performance of 2.6B SDXL-DMD2 (Yin et al., 2024) and SD1.5-DPO (Wallace et al., 2024). These results establish Diff-Instruct* as a versatile, architecture-flexible approach for aligning one-step text-to-image generators with human preferences.

2 PRELIMINARY

Diffusion Models. In this section, we introduce preliminary knowledge and notations about diffusion models. Assume we observe data from the underlying distribution $q_d(\boldsymbol{x})$. The goal of generative modeling is to train models to generate new samples $\boldsymbol{x} \sim q_0(\boldsymbol{x})$. Under mild conditions, the forward diffusion process of a diffusion model can transform initial distribution q_0 towards some simple noise distribution,

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{F}(\boldsymbol{x}_t, t)\mathrm{d}t + G(t)\mathrm{d}\boldsymbol{w}_t, \tag{2.1}$$

where $\boldsymbol{F}$ is a pre-defined vector-valued drift function, $G(t)$ is a pre-defined scalar-value diffusion coefficient, and $\boldsymbol{w}_t$ denotes an independent Wiener process. A continuous-indexed score network $\boldsymbol{s}_\varphi(\boldsymbol{x}, t)$ is employed to approximate marginal score functions of the forward diffusion process (2.1). The learning of score networks is achieved by minimizing a weighted denoising score matching objective (Vincent, 2011; Song et al., 2020),

$$\mathcal{L}_{DSM}(\varphi) = \int_{t=0}^{T} \lambda(t)\mathbb{E}_{\substack{\boldsymbol{x}_0\sim q_0,\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} \|\boldsymbol{s}_\varphi(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\|_2^2 \mathrm{d}t. \tag{2.2}$$

Here the weighting function $\lambda(t)$ controls the importance of the learning at different time levels and $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ denotes the conditional transition of the forward diffusion (2.1). After training, the score network $\boldsymbol{s}_\varphi(\boldsymbol{x}_t, t) \approx \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t)$ is a good approximation of the marginal score function of the diffused data distribution.

3 PREFERENCE ALIGNMENT USING DIFF-INSTRUCT*

In this section, we introduce Diff-Instruct*, a general method tailored for training one-step text-to-image generator according to human preference. We begin with outlining the problem setup and

defining notations. We frame the training objective as a special form of reinforcement learning using human feedback (RLHF). We then introduce a family of score-based probability divergences and show how these divergences effectively regularize the RLHF process to align closely with human aesthetics and content preferences. Next, we introduce the classifier-free reward and discuss how to balance the explicit human reward model and the implicit classifier-free reward.

Problem Setup. Assume we have a human reward function $r(\boldsymbol{x}_0, \boldsymbol{c})$, which encodes the human preference for an image $\boldsymbol{x}_0$ and corresponding text description $\boldsymbol{c}$. Besides, we also have a pre-trained diffusion model which will later act as a reference distribution $p_{ref}(\boldsymbol{x}_0) = q_0(\boldsymbol{x}_0)$. We assume the reference diffusion model is specified by the score function

$$\boldsymbol{s}_{q_t}(\boldsymbol{x}_t) \coloneqq \nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t) \tag{3.1}$$

where $q_t(\boldsymbol{x}_t)$'s is the underlying distribution diffused at time t according to (2.1). We assume that the pre-trained diffusion model well captures the ground-truth data distribution, and thus will be the only item of consideration for our approach.

Our goal is to train a human-preferred one-step generator model g_θ that generates images by directly mapping a random noise $\boldsymbol{z} \sim p_z$ to obtain $\boldsymbol{x}_0 = g_\theta(\boldsymbol{z}, \boldsymbol{c})$, conditioned on the input text $\boldsymbol{c} \sim \mathcal{C}$. The generator's output distribution, $p_\theta(\boldsymbol{x}_0|\boldsymbol{c})$, should maximize the expected human rewards while adhering to the constraints on the divergence to the reference distribution $p_{ref}(\cdot)$. Let $\boldsymbol{D}(\cdot,\cdot)$ be a distribution divergence. For any fixed prompt $\boldsymbol{c}$, the training objective is defined as:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{x}_0 \sim p_\theta(\boldsymbol{x}_0|\boldsymbol{c})}\big[r(\boldsymbol{x}_0, \boldsymbol{c})\big], \quad \text{s.t. } \boldsymbol{D}(p_\theta(\cdot|\boldsymbol{c}), p_{ref}(\cdot|\boldsymbol{c})) \le \delta \tag{3.2}$$

The constraint $\boldsymbol{D}(p_\theta(\cdot|\boldsymbol{c}), p_{ref}(\cdot|\boldsymbol{c})) \le \delta$ is often referred to as a *trust-region* in literature of reinforcement learning (Schulman, 2015; Schulman et al., 2017). This objective (3.2) is equivalent to the minimization of objective (3.3) that adds the constraining term onto the negative reward:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}_0 \sim p_\theta(\boldsymbol{x}_0|\boldsymbol{c})}\big[-\alpha r(\boldsymbol{x}_0, \boldsymbol{c})\big] + \boldsymbol{D}(p_\theta, p_{ref}) \tag{3.3}$$

Here α is a coefficient that balances reward influences and $\boldsymbol{D}(\cdot)$ acts as a regularization term. Traditional reinforcement learning from human feedback (RLHF) methods in large language models (Ouyang et al., 2022) use the Kullback-Leibler divergences for regularization. Recent work (Luo, 2024) has studied using the integral of KL divergence in objective (3.3) to train one-step text-to-image generators. However, since the KL divergences are defined with the density ratio of two distributions, any misalignment of density supports will lead to severe numerical instability, potentially resulting in annoying mode-seeking behavior (Bishop, 2006). Besides, some recent works have shown that score-based divergences (Zhou et al., 2024b; Luo et al., 2024c) result in better and more stable performance than KL divergences (Luo et al., 2024b; Yin et al., 2023; Nguyen & Tran, 2023) in the literature of diffusion distillation. Motivated by the above two inspirations, we propose a novel score-based online PPO algorithm that uses a general family of score-based divergence as regularization in this paper. We provide solid theoretical foundations of score-based regularization and its better empirical performances through large-scale text-to-image models.

3.1 General Score-based Divergences

Different from KL divergence, we can define the regularization term $\boldsymbol{D}(p_\theta, p_{ref})$ via the following general score-based divergence. Assume $\mathbf{d} : \mathbb{R}^d \to \mathbb{R}$ is a scalar-valued proper distance function (i.e., a non-negative function that satisfies $\forall \boldsymbol{x}, \mathbf{d}(\boldsymbol{x}) \ge 0$ and $\mathbf{d}(\boldsymbol{x}) = 0$ if and only if $\boldsymbol{x} = \mathbf{0}$). Given a parameter-independent sampling distribution π_t that has large distribution support, we can formally define a time-integral score divergence as

$$\mathbf{D}^{[0,T]}(p_\theta, p_{ref}) \coloneqq \int_{t=0}^{T} w(t) \mathbb{E}_{\boldsymbol{x}_t \sim \pi_t} \Big\{ \mathbf{d}(\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)) \Big\} \mathrm{d}t, \tag{3.4}$$

where $p_{\theta,t}$ and q_t denote the marginal densities of the diffusion process (2.1) at time t initialized with $p_{\theta,0} = p_\theta$ and $q_0 = p_{ref}$ respectively. $w(t)$ is an integral weighting function. Clearly, we have $\mathbf{D}^{[0,T]}(p_\theta, p_{ref}) = 0$ if and only if $p_\theta(\boldsymbol{x}_0) = p_{ref}(\boldsymbol{x}_0),\ a.s.\ \pi_0$.

Algorithm 1: Diff-Instruct* for training human-preferred one-step text-to-image generators.

Input: prompt dataset $\mathcal{C}$, generator $g_\theta(\boldsymbol{x}_0|\boldsymbol{z},\boldsymbol{c})$, prior distribution p_z, reward model $r(\boldsymbol{x},\boldsymbol{c})$, reward model scale α_{rew}, CFG reward scale α_{cfg}, reference diffusion model $\boldsymbol{s}_{ref}(\boldsymbol{x}_t|c,\boldsymbol{c})$, assistant diffusion $\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})$, forward diffusion $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$ (2.1), assistant diffusion updates rounds K_{TA}, time distribution $\pi(t)$, diffusion model weighting $\lambda(t)$, generator loss time weighting $w(t)$.

while *not converge* **do**

freeze θ, update ψ for K_{TA} rounds using SGD by minimizing

$$\mathcal{L}(\psi) = \mathbb{E}_{\substack{\boldsymbol{c}\sim\mathcal{C},\boldsymbol{z}\sim p_z,t\sim\pi(t)\\ \boldsymbol{x}_0=g_\theta(\boldsymbol{z}|\boldsymbol{c}),\boldsymbol{x}_t|\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} \lambda(t)\|\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c}) - \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\|_2^2 \mathrm{d}t.$$

freeze ψ, update θ using SGD by minimizing loss

$$\begin{aligned}\mathcal{L}_{DI*}(\theta) =& \mathbb{E}_{\substack{\boldsymbol{c}\sim\mathcal{C},\boldsymbol{z}\sim p_z,\\ \boldsymbol{x}_0=g_\theta(\boldsymbol{z},\boldsymbol{c})}}\Bigg\{-\alpha_{rew}\cdot r(\boldsymbol{x}_0,\boldsymbol{c}) + \mathbb{E}_{\substack{t\sim\pi(t),\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\Bigg[\\ &- w(t)\big\{\mathbf{d}'(\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c}) - \boldsymbol{s}_{ref}(\boldsymbol{x}_t|t,\boldsymbol{c}))\big\}^T\big\{\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c}) - \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\big\}\\ &+ \alpha_{cfg}\cdot w(t)\big\{\boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t,\boldsymbol{c}) - \boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t,\boldsymbol{\varnothing})\big\}^T\boldsymbol{x}_t\Bigg]\Bigg\}\end{aligned} \tag{3.7}$$

end

return θ,ψ.

3.2 Diff-Instruct*

Recall that g_θ is a one-step model, therefore samples from p_θ can be implemented through a direct mapping $\boldsymbol{x}_0 = g_\theta(\boldsymbol{z}|\boldsymbol{c})$. With the score-based regularization term (3.4), for each given text prompt $\boldsymbol{c}$, we can formally write down our training objective to minimize as:

$$\mathcal{L}_{Orig}(\theta) = \mathbb{E}_{\substack{\boldsymbol{z}\sim p_z,\\ \boldsymbol{x}_0=g_\theta(\boldsymbol{z},\boldsymbol{c})}}\big[-\alpha r(\boldsymbol{x}_0,\boldsymbol{c})\big] + \mathbf{D}^{[0,T]}(p_\theta, p_{ref}) \tag{3.5}$$

Now we are ready to reveal the objective of Diff-Instruct* that we use to train human-preferred one-step generator g_θ. Notice that directly minimizing objective (3.5) is intractable because we do not know the relationship between θ and corresponding $p_{\theta,t}$. However, we show in Theorem 3.1 that an equivalent tractable loss (3.6) will have the same θ gradient as the intractable loss function (3.5):

$$\begin{aligned}\mathcal{L}_{DI*}(\theta) =& \mathbb{E}_{\substack{\boldsymbol{z}\sim p_z,\\ \boldsymbol{x}_0=g_\theta(\boldsymbol{z})}}\Bigg[-\alpha r(\boldsymbol{x}_0,\boldsymbol{c})\\ &+ \int_{t=0}^T w(t)\mathbb{E}_{\substack{\boldsymbol{x}_t|\boldsymbol{x}_0\\ \sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\Big\{-\mathbf{d}'(\boldsymbol{y}_t)\Big\}^T\Big\{\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\Big\}\mathrm{d}t\Bigg]\end{aligned} \tag{3.6}$$

with $\boldsymbol{y}_t := \boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)$.

Theorem 3.1. Under mild assumptions, if we take the sampling distribution in (3.4) as $\pi_t = p_{\mathrm{sg}[\theta],t}$, then the θ gradient of (3.5) is the same as the objective (3.6): $\frac{\partial}{\partial\theta}\mathcal{L}_{Orig}(\theta) = \frac{\partial}{\partial\theta}\mathcal{L}_{DI*}(\theta)$.

We will give the proof in Appendix A.1. In practice, we can use another assistant diffusion model $\boldsymbol{s}_\psi(\boldsymbol{x}_t,t)$ to approximate the generator model's score function $\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t)$ pointwise, which was also done in the literature of diffusion distillations works such as Zhou et al. (2024a;b); Luo et al. (2024b;c); Yin et al. (2023; 2024). Therefore, we can alternate between 1) updating the assistant diffusion $\boldsymbol{s}_\psi(\boldsymbol{x}_t,t)$ using generator-generated samples (which are efficient) and 2) updating the generator by minimizing the tractable objective (3.6). We name our training method that minimizes the objective $\mathcal{L}_{DI*}(\theta)$ in (3.6) the Diff-Instruct* because it is inspired by Diff-Instruct(Luo et al., 2024b) and Diff-Instruct++(Luo, 2024) that involves an additional diffusion model and a reward model to train one-step generators.

3.3 Decoupling the Explicit Reward and Implicit Guidance-based Rewards

Classifier-free Guidance Corresponds to Implicit Reward. In previous sections, we have shown in theory that with explicitly available reward models, we can readily train the one-step generator to

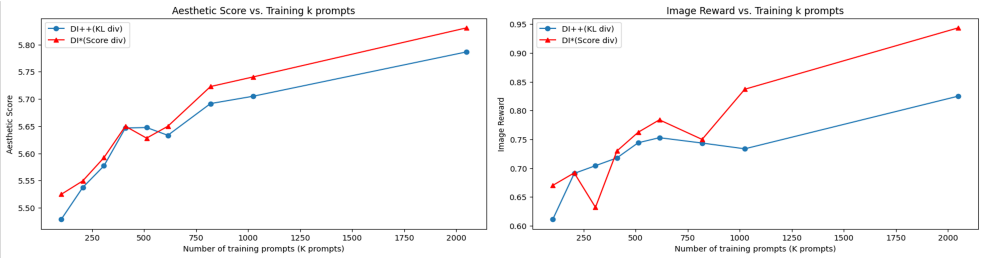

Figure 3: Comparison of Aesthetic Scores and Image Reward on 1K MSCOCO 2017 validation prompts of Score-based (DI*) and KL divergence (DI++(Luo, 2024)) for alignment with (3.5).

align with human preference. In this section, we enhance the DI* by incorporating the classifier-free reward that is implied by the classifier-free guidance of diffusion models.

The classifier-free guidance (Ho & Salimans, 2022) (CFG) uses a modified score function of a form

$$\widetilde{\boldsymbol{s}}_{ref}(\boldsymbol{x}_t, t|\boldsymbol{c}) := \boldsymbol{s}_{ref}(\boldsymbol{x}_t, t|\boldsymbol{\varnothing}) + \omega\big\{\boldsymbol{s}_{ref}(\boldsymbol{x}_t, t|\boldsymbol{c}) - \boldsymbol{s}_{ref}(\boldsymbol{x}_t, t|\boldsymbol{\varnothing})\big\}$$

to replace the original conditions score function $\boldsymbol{s}_{ref}(\boldsymbol{x}_t, t|\boldsymbol{c})$. Using CFG for diffusion models empirically leads to better sampling quality.

As is first pointed out by Luo (2024), the classifier-free guidance is related to an implicit reward function. In this part, we derive a tractable loss function that minimizes the so-called classifier-free reward, which we use together with the explicit reward $r(\cdot, \cdot)$ in DI*.

Theorem 3.2. Under mild conditions, if we set an implicit reward function as (3.9), the loss (3.8)

$$\mathcal{L}_{cfg}(\theta) = \int_{t=0}^{T} \mathbb{E}_{\substack{\boldsymbol{z}\sim p_{\boldsymbol{z}}, \boldsymbol{x}_0 = g_\theta(\boldsymbol{z}, \boldsymbol{c}) \\ \boldsymbol{x}_t|\boldsymbol{x}_0 \sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}} w(t) \Big\{ \boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t, \boldsymbol{c}) - \boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t, \boldsymbol{\varnothing}) \Big\}^T \boldsymbol{x}_t \mathrm{d}t \tag{3.8}$$

has the same gradient as the negative implicit reward function (3.9)

$$-r(\boldsymbol{x}_0, \boldsymbol{c}) = -\int_{t=0}^{T} \mathbb{E}_{\boldsymbol{x}_t \sim p_{\theta,t}} w(t) \log \frac{p_{ref}(\boldsymbol{x}_t|t, \boldsymbol{c})}{p_{ref}(\boldsymbol{x}_t|t)} \mathrm{d}t. \tag{3.9}$$

The notation $\mathrm{sg}[\boldsymbol{x}_t]$ means detaching the θ gradient on $\boldsymbol{x}_t$. We give the proof of Theorem 3.2 in Appendix A.2. Theorem 3.2 gives a tractable loss function (3.8) aiming to minimize the negative classifier-free reward function. Therefore, we can scale and add this loss $\mathcal{L}_{cfg}(\theta)$ (3.8) to the DI* loss (3.6) to balance the effects of explicit reward and implicit CFG reward.

Remark 3.3. Theorem 3.2 has decoupled the classifier-free reward with the explicit reward. This distinguishes DI* from previous pure diffusion distillation methods such as Zhou et al. (2024a) that incorporated classifier-free guidance when computing the score functions without decoupling it with the conditional score functions. On the contrary, our theory on decoupling explicit and implicit rewards shows a new perspective and provides flexibility in training models for different needs.

3.4 The Practical Algorithm

Now it is time for us to introduce the practical algorithm. As Algorithm 1 (and a more executable version in Algorithm 2) shows, the DI* involves three models, with one generator model g_θ, one reference diffusion model $\boldsymbol{s}_{ref}$ and one assistant diffusion model $\boldsymbol{s}_\psi$. The reference diffusion does not need to be trained, while the generator and the assistant diffusion are updated alternatively. Two hyper-parameters, the α_{rew} and α_{cfg} control the strength of the explicit reward and the implicit CFG reward during training. The explicit reward model can either be an off-the-shelf reward model, such as the CLIP similarity score (Radford et al., 2021), or the Image Reward (Xu et al., 2023a) or trained in-house with researchers' internal human feedback data. Due to page limitations, we put a discussion about the meanings of hyper-parameters in Appendix B.2.

Flexibility in Distance Functions. Clearly, various choices of distance function $\mathbf{d}(.)$ result in different training algorithms. For instance, $\mathbf{d}(\boldsymbol{y}_t) = \|\boldsymbol{y}_t\|_2^2$ is a naive choice. Interestingly, such a

distance function has been studied in pure diffusion distillation literature in Zhou et al. (2024b;a); Luo et al. (2024c). In this paper, we draw inspiration from (Luo et al., 2024c) and find that using the so-called pseudo-Huber distance leads to better performance. The distance and corresponding loss writes $\boldsymbol{d}(\boldsymbol{y}) \coloneqq \sqrt{\|\boldsymbol{y}_t\|_2^2 + c^2} - c$, and

$$\mathbf{D}^{[0,T]}(p_\theta, p_{ref}) = -\left\{ \frac{\boldsymbol{y}_t}{\sqrt{\|\boldsymbol{y}_t\|_2^2 + c^2}} \right\}^T \left\{ \boldsymbol{s}_\psi(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \right\}. \tag{3.10}$$

Here $\boldsymbol{y}_t \coloneqq \boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)$.

Flexibility in Reference Diffusion and Generators. Another advantage of DI* is the flexible choice of reference diffusion models and generator architecture. DI* can take currently the best diffusion models as reference, benefitting from the rapidly growing diffusion community. Besides, the generator can be flexible across GAN generators, diffusion networks, as well as 3D Neural Radiance Fields (Mildenhall et al., 2021) which has been used for text-to-3D generation using text-to-2D diffusion models (Poole et al., 2022). We put detailed discussions in Appendix B.3.

4 Related Works

Diffusion Distillation Through Divergence Minimization. Diff-Instruct* is inspired by research on diffusion distillation (Luo, 2023) which aims to minimize certain distribution divergence to train one-step generators. Luo et al. (2024b) first study the diffusion distillation by minimizing the Integral KL divergence. Yin et al. (2023) generalize such a concept and add a data regression loss for better performance. Zhou et al. (2024b) study the distillation by minimizing the Fisher divergence. Luo et al. (2024c) study the distillation using the general score-based divergence. Many other works also introduced additional techniques and improved the performance (Geng et al., 2023; Kim et al., 2023; Song et al., 2023; Song & Dhariwal; Nguyen & Tran, 2023; Song et al., 2024; Yin et al., 2024; Zhou et al., 2024a; Heek et al., 2024; Xie et al., 2024; Salimans et al., 2024; Geng et al., 2024).

Preference Alignment for Diffusion Models and One-step Generators. In recent years, many works have emerged trying to align diffusion models with human preferences. There are three main lines of alignment methods for diffusion models. 1) The first kind of method fine-tunes the diffusion model over a specifically curated image-prompt dataset (Dai et al., 2023; Podell et al., 2023). 2) the second line of methods tries to maximize some reward functions either through the multi-step diffusion generation output (Prabhudesai et al., 2023; Clark et al., 2023; Lee et al., 2023) or through policy gradient-based RL approaches (Fan et al., 2024; Black et al., 2023). For these methods, the backpropagation through the multi-step diffusion generation output is expensive and hard to scale. 3) the third line, such as Diffusion-DPO (Wallace et al., 2024), Diffusion-KTO (Yang et al., 2024), tries to directly improve the diffusion model's human preference property with raw collected data instead of reward functions. Besides the human preference alignment of diffusion models, Diff-Instruct++(Luo, 2024) recently arose as the first attempt to improve human preferences for one-step generators. Though inspired by DI++, DI* uses score-based divergences which are technically different from the KL divergence used in DI++. Besides, as we show in Section 5.2, DI* archives better performances than DI++ in Luo (2024).

5 Experiments

In this section, we train one-step text-to-image generators of two kinds of architectures: UNet-based generators that use the Stable Diffusion 1.5 (Rombach et al., 2022) as the reference diffusion, and DiT-based generators that use the PixelArt-α as the reference diffusion. For both experiments, we use the off-the-shelf Image Reward (Xu et al., 2023a) as our explicit reward model, and we also incorporate the implicit classifier-free reward as we have discussed in Section 3.3. With both architectures, one-step models trained with DI* show stronger human preference performances than previous models with large margins.

5.1 Experiment Setttings

Experiment Settings for SD1.5 and SDXL Experiment. For experiments of SD1.5, we use the open-sourced SD1.5 of a resolution of 512×512 as our reference diffusion in Algorithm 1. We

Table 1: Quantitative comparisons of text-to-image models on **MSCOCO-2017 validation** prompts (the upper part) and **Parti(Yu et al., 2022)** Prompts (the under part). DI* is short for Diff-Instruct*. α_r and α_c are short for α_{rew} and α_{cfg} in Algorithm 1. † means our implementation. Data means the model needs image data for training. Sampling means the model needs to draw samples from reference diffusion models. Reward means the model needs a human reward model for training. † indicates our implementation. ‡ indicates the same 4-step model of DMD2 but with different inference steps.

Model	Steps	Type	Params	Image Reward	Aes Score	Pick Score	CLIP Score	Additional Requirements
SD15-DPO(Wallace et al., 2024)	15	UNet	0.86B	0.20	5.29	0.214	31.07	Data, Sampling
SD15-DPO(Wallace et al., 2024)	25	UNet	0.86B	0.28	5.37	0.218	31.25	Preference Data
SD15-LCM(Luo et al., 2023a)	1	UNet	0.86B	-1.58	5.04	0.194	27.20	Data, Sampling
SD15-LCM(Luo et al., 2023a)	4	UNet	0.86B	-0.23	5.40	0.214	30.11	Data, Sampling
SD15-TCD(Zheng et al., 2024)	1	UNet	0.86B	-1.49	5.10	0.196	28.30	Data, Sampling
SD15-TCD(Zheng et al., 2024)	4	UNet	0.86B	-0.04	5.28	0.212	30.43	Data, Sampling
PeRFlow(Yan et al., 2024)	4	UNet	0.86B	-0.20	5.51	0.211	29.54	Data, Sampling
SD15-Hyper(Ren et al., 2024)	1	UNet	0.86B	0.28	5.49	0.214	30.82	Data, Sampling
SD15-Hyper(Ren et al., 2024)	4	UNet	0.86B	0.42	5.41	0.217	31.03	Reward, Seg-Model
SD15-InstaFlow(Liu et al., 2023)	1	UNet	0.86B	-0.16	5.03	0.207	30.68	Data, Sampling
SDXL-Base(Rombach et al., 2022)	25	UNet	2.6B	0.74	5.57	0.226	31.83	Image-Text
SDXL-DMD2‡-1024(Yin et al., 2024)	1	UNet	2.6B	0.82	5.45	0.224	31.78	Image-Text
SDXL-DMD2‡-1024(Yin et al., 2024)	4	UNet	2.6B	0.87	5.52	**0.231**	31.50	Image-Text
SDXL-DMD2‡-512(Yin et al., 2024)	1	UNet	2.6B	0.36	5.03	0.215	31.54	Image-Text
SDXL-DMD2‡-512(Yin et al., 2024)	4	UNet	2.6B	-0.18	5.17	0.206	29.28	Image-Text
SD15-DMD2-512(Yin et al., 2024)	1	UNet	2.6B	-0.12	5.24	0.211	30.00	Image-Text
SD21-Turbo(Sauer et al., 2023b)	1	UNet	0.86B	0.56	5.47	0.225	31.50	Image-Text
SD15-Base(Rombach et al., 2022)	15	UNet	0.86B	0.08	5.25	0.212	30.99	Image-Text
SD15-Base(Rombach et al., 2022)	25	UNet	0.86B	0.22	5.32	0.216	31.13	Image-Text
PixelArt-α-512(Chen et al., 2023)	25	DiT	0.6B	0.82	6.01	0.227	31.20	Image-Text
PixelArt-α-512(Chen et al., 2023)	15	DiT	0.6B	0.82	6.03	0.226	31.16	Image-Text
SD15-SiDLSG(Zhou et al., 2024a)	1	UNet	0.86B	-0.18	5.16	0.210	30.04	Text
SDXL-DMD2-1024(Yin et al., 2024)	1	UNet	2.6B	0.85	5.46	0.225	31.86	Image-Text
SD15-DI++(Luo, 2024)†	1	UNet	0.86B	0.82	5.78	0.219	30.30	Reward
DiT-DI++(Luo, 2024)†	1	DiT	0.6B	1.24	6.19	0.225	30.80	Reward
SD15-DI*($\alpha_r = 0, \alpha_c = 1.5$)	1	UNet	0.86B	0.34	5.27	0.217	30.83	Reward
SD15-DI*($\alpha_r = 100, \alpha_c = 1.5$)	1	UNet	0.86B	0.62	5.44	0.218	30.76	Reward
SD15-DI*($\alpha_r = 1000, \alpha_c = 4.5$)	1	UNet	0.86B	0.73	5.56	0.219	30.71	Reward
SD15-DI*($\alpha_r = 1000, \alpha_c = 1.5$)	1	UNet	0.86B	0.94	5.83	0.220	30.49	Reward
SDXL-DI*-1024($\alpha_r = 1000, \alpha_c = 8.0$)	1	UNet	2.6B	0.88	5.56	0.225	**32.07**	Reward
DiT-DI*($\alpha_r = 1, \alpha_c = 4.5$)	1	DiT	0.6B	0.98	6.02	0.225	31.00	Reward
DiT-DI*($\alpha_r = 10, \alpha_c = 4.5$)	1	DiT	0.6B	**1.31**	**6.30**	0.225	30.84	Reward
SDXL-Base(Rombach et al., 2022)	15	UNet	2.6B	0.69	5.68	0.224	32.76	Image-Text
PixelArt-α-512(Chen et al., 2023)	15	DiT	0.6B	0.96	6.00	**0.227**	31.76	Image-Text
SDXL-DMD2-1024(Yin et al., 2024)	1	UNet	2.6B	0.94	5.53	0.225	33.00	Image-Text
SDXL-DI*-1024($\alpha_r = 1000, \alpha_c = 8.0$)	1	UNet	2.6B	1.06	5.61	0.225	**33.27**	Reward
DiT-DI*($\alpha_r = 1, \alpha_c = 4.5$)	1	DiT	0.6B	0.96	6.06	0.224	31.11	Reward
DiT-DI*($\alpha_r = 10, \alpha_c = 4.5$)	1	DiT	0.6B	**1.26**	**6.23**	0.225	30.64	Reward

implement our experiments based on SiD-LSG (Zhou et al., 2024a) codebase. We construct the one-step generator with the same architecture as the reference SD1.5 model, following the same configuration of SiD-LSG. We use the prompts of the LAION-AESTHETIC dataset with an aesthetic score larger than 6.25, which resulting a total of 3M text prompts. To better explore the advantages of our score-based divergence over KL divergence, we refer to a recent work (Luo, 2024) that uses KL divergence for training and conducts a detailed comparison between two divergences. We explore different combinations and find that $(\alpha_{rew} = 1000, \alpha_{cfg} = 1.5)$ in Algorithm 1 is the best. For SD1.5 experiments, we find that training one-step generators from scratch leads to longer training, therefore we initialize our generator with the weights of the SiD-LSG pre-trained one-step model. For experiments of SDXL, we use the SDXL(Podell et al., 2023) as the teacher model and its architecture as the one-step student model. We initialize the one-step generator with the pre-trained DMD2-SDXL-1step model (Yin et al., 2024), which is a pretty good initialization based on SDXL architectures. We follow similar settings as the SD1.5 experiment: using a scale for explicit ImageReward of 1000 and a scale for implicit CFG reward of 8.0.

Experiment Settings for PixelArt-α Experiment. PixArt-α is a high-quality DiT-based(Peebles & Xie, 2022) open-sourced text-to-image diffusion model. We use the DiT architecture and the 0.6B PixArt-α of a resolution of 512×512 as our reference diffusion to demonstrate the compatibil-

ity of DI* for different neural network architectures. We use the prompts from the training dataset of PixArt-α (the SAM-Recaptioned Dataset), resulting in a total of 10M prompts. After exploring different combinations, we find that $(\alpha_{rew} = 10, \alpha_{cfg} = 4.5)$ in Algorithm 1 gives the best performances. For DiT generators with a total of 0.6B parameters, we find that training from scratch without initialization results in strong results. Therefore we do not apply special initialization for DiT-based one-step generators. Our best DiT-DI* model in Table 2 is trained from scratch with 8 H100-80G GPUs for 200000 iterations with a batch size of 128. The total wall-clock training costs is less than 30 hours. As a comparison, other industry models in Table 1 usually require hundreds of A100 GPU days for training. Such a low cost might benefit from the property that DI* needs neither real image data nor samples from reference diffusion models.

Quantitative Evaluations Metrics. We compare our generators with other leading open-sourced models that are either based on SD1.5 diffusion models or larger models such as SDXL. For all models, we compute four standard scores in Table 1: the Image Reward (Xu et al., 2023a), the Aesthetic Score (Schuhmann, 2022), the PickScore(Kirstain et al., 2023), and the CLIP score(Radford et al., 2021) on the same 1K prompts randomly sampled from COCO-2017-validation(Lin et al., 2014) set on the same computing devices. We have compared the results on 1K prompts and 30K COCO prompts and found similar results. Since our training prompts do not involve COCO validation prompts, the evaluation of COCO prompts can be viewed as an out-of-training set evaluation. We also evaluate models with Human Preference Score v2.0 (HPSv2.0) (Wu et al., 2023) in Table 2. The HPSv2.0 is a standard score for evaluating models' human preferences across different styles.

5.2 Performances and Findings

Quantitative Comparison: DI* Achieves SoTA Human Preference Scores. As Table 1 shows, our best 0.6B DiT-DI* model outperforms all other open-sourced models. It achieves a COCO (out-of-sample) Image Reward of **1.31**, which is **50%** better than the second best 2.6B SDXL-DMD2-4Step model of 1024 resolution. It also outperforms the SDXL model with a margin of **70%**. It also shows an Aesthetic Score of **6.30**, which is **14.3%** better than the 2.6B SDXL-DMD2-4Step model. Among SD1.5-based models, our best SD1.5-DI* model outperforms other models with significant margins. This result demonstrates the compatibility of DI* across UNet and DiT architectures.

As Table 2 shows, our 0.6B DiT-DI* one-step model achieves a record-breaking HPSv2.0 score of **28.70** across open-sourced text-to-image models. It clearly outperforms the autoregressive models such as 6B CogView2 (Ding et al., 2022), diffusion models such as 5.5B Dalle-E 2 (Ramesh et al., 2022), 5B GLIDE(Nichol et al., 2021), 2.6B SDXL(Podell et al., 2023), and 4.3B DeepFloyd-XL with up to 30+ generation steps. Such a lightweight, high-performance, and high-efficiency model will bring big impacts on applications that need real-time generations.

Qualitative Comparisons. As Figure 1 and Figure 2 show, models trained with DI* show better layouts, richer colors, and more aesthetic face preferences. We find such an advantage is stronger for scene generations. Figure 4 shows a visualization of the SD1.5-DI* model in Table 2 before and after alignments with DI*. It shows that models after alignment produce images with richer colors. We also evaluate the COCO-FID values of SD1.5-based one-step models with and without preference alignment with DI*. The FID of the alignment model is 18.44, while the un-alignment model (the SiD-LSG which we use as the initialization of the one-step generator) has an FID of 8.27. This phenomenon suggests that only using ImageReward(which tends to be subjective) to align models with DI* can potentially harm objective metrics such as FID and CLIPScore. A naive solution to hack CLIPScore is to simply include the CLIP score and PickScore as two other sources of reward when using DI*, which will definitely improve these scores. However, since we want to verify the generalization ability of one reward to others using DI*, we only use the ImageReward and observed other scores in this paper.

Score-based Divergence is Better than KL Divergence. In Figure 3 we compare the use of score-based divergence of DI* and the KL divergence of DI++(Luo, 2024) for RLHF regularization in (3.5). We fix the $\alpha_{rew} = 1000$ and $\alpha_{cfg} = 1.5$ for both DI++ and DI* and use the same Image Reward as an explicit reward model for training the SD1.5-based one-step generators. As we can see in Figure 3, for each iteration, DI* has a better score than DI++. Besides, DI* achieves the best final results. The reason for the worse performance of KL divergence might be its definition, which

involves the ratio of two distributions, which may lead to unstable numerical performances when two distributions have misaligned density supports.

Models Trained with One Reward can Generalize to Other Human Preference Scores. Another plausible property of DI* is that the one-step model trained with one human preference reward can be generalized to other human preference scores. For instance, in our experiments, we use the Image Reward as the explicit reward. However, after training with DI*, the model shows strong Aesthetic Scores and HPSv2.0 scores. This property makes it possible for researchers can pre-trained very strong human preference reward models with domain-specific human feedback data, then they train one-step models with that reward and will get strong performances under other metrics.

6 Conclusion and Limitations

In this paper, we present Diff-Instruct*, a novel approach for aligning human-preferred one-step text-to-image generators. By formulating the training objective as a maximization of expected human reward functions with a score-based divergence regularization, we have developed practical losses and easy-to-implement algorithms. Our results show that DiT-based one-step text-to-image generators trained with DI* achieve new state-of-the-art human preference performances.

Nonetheless, DI* has its limitations. We empirically find three typical mistakes the generator often makes. (1) The Generator Sometimes Generates Bad Human Faces and Hands. (2) The aligned model still can not count correctly. Figure 5 shows some occasional bad generation cases. We believe that consistently improving both the generator architecture and the reward models will lead to better models. Besides, we are also interested in following directions that call for further research. First, in this paper, we only study using only one reward model to train the generator. However, training generators with multiple rewards is still unexplored. Second, the DI* needs a pre-trained reward model. However, methods like DPO (Rafailov et al., 2024; Wallace et al., 2024) have put a new setup that trains generative models directly using human feedback data. Whether it is possible to develop DPO-like algorithms based on DI* is a promising research direction. Third, it is also interesting to explore training multistep models instead of one-step generators for better performance. We hope these future directions could contribute more to the community.

Table 2: **HPSv2.0** (upper table) and **HPSv2.1** (under table). We compare open-sourced models regardless of their base model and architecture. † indicates our implementation. ‡ indicates the same 4-step model of DMD2 but with different inference steps.

Model	Animation	Concept-art	Painting	Photo	Average
GLIDE (Nichol et al., 2021)	23.34	23.08	23.27	24.50	23.55
LAFITE (Zhou et al., 2022)	24.63	24.38	24.43	25.81	24.81
VQ-Diffusion (Gu et al., 2022)	24.97	24.70	25.01	25.71	25.10
FuseDream (Liu et al., 2021)	25.26	25.15	25.13	25.57	25.28
Latent Diffusion (Rombach et al., 2022)	25.73	25.15	25.25	26.97	25.78
CogView2 (Ding et al., 2022)	26.50	26.59	26.33	26.44	26.47
DALL·E mini	26.10	25.56	25.56	26.12	25.83
Versatile Diffusion (Xu et al., 2023b)	26.59	26.28	26.43	27.05	26.59
VQGAN + CLIP (Esser et al., 2021)	26.44	26.53	26.47	26.12	26.39
DALL·E 2 (Ramesh et al., 2022)	27.34	26.54	26.68	27.24	26.95
Stable Diffusion v1.4 (Rombach et al., 2022)	27.26	26.61	26.66	27.27	26.95
Stable Diffusion v2.0 (Rombach et al., 2022)	27.48	26.89	26.86	27.46	27.17
Epic Diffusion	27.57	26.96	27.03	27.49	27.26
DeepFloyd-XL	27.64	26.83	26.86	27.75	27.27
Openjourney	27.85	27.18	27.25	27.53	27.45
MajicMix Realistic	27.88	27.19	27.22	27.64	27.48
ChilloutMix	27.92	27.29	27.32	27.61	27.54
Deliberate	28.13	27.46	27.45	27.62	27.67
Realistic Vision	28.22	27.53	27.56	27.75	27.77
SDXL-base(Podell et al., 2023)	28.42	27.63	27.60	27.29	27.73
SDXL-Refiner(Podell et al., 2023)	28.45	27.66	27.67	27.46	27.80
Dreamlike Photoreal 2.0	28.24	27.60	27.59	27.99	27.86
SD15-DPO-15Step(Wallace et al., 2024)	27.11	26.75	26.70	27.30	26.97
SD15-DPO-25Step(Wallace et al., 2024)	27.54	26.97	26.99	27.49	27.25
SD15-LCM-1Step(Luo et al., 2023a)	23.35	23.41	23.53	23.81	23.52
SD15-LCM-4Step(Luo et al., 2023a)	26.42	25.79	25.95	26.91	26.27
SD15-TCD-1Step(Zheng et al., 2024)	23.37	23.16	23.26	23.88	23.42
SD15-TCD-4Step(Zheng et al., 2024)	26.67	26.25	26.26	27.19	26.59
SD15-Hyper-1Step(Ren et al., 2024)	27.76	27.36	27.41	27.63	27.54
SD15-Hyper-4Step(Ren et al., 2024)	28.04	27.39	27.42	27.89	27.69
SD15-Instaflow-1Step(Liu et al., 2023)	26.07	25.80	25.89	26.32	26.02
SD15-PeReflow-1Step(Yan et al., 2024)	25.70	25.45	25.57	25.96	25.67
SD15-BOOT-1Step(Gu et al., 2023)	25.29	24.40	24.61	25.16	24.86
SD21-SwiftBrush-1Step(Nguyen & Tran, 2023)	26.91	26.32	26.37	27.21	26.70
SD21-TURBO-1Step(Sauer et al., 2023b)	27.48	26.86	27.46	26.89	27.71
SDXL-DMD2‡-1Step-1024(Yin et al., 2024)	27.67	27.02	27.01	26.94	27.16
SDXL-DMD2‡-4Step-1024(Yin et al., 2024)	28.97	27.99	27.90	28.28	28.29
SDXL-DMD2‡-1Step-512(Yin et al., 2024)	27.70	27.07	27.02	26.94	27.18
SDXL-DMD2‡-4Step-512(Yin et al., 2024)	27.22	26.65	26.62	26.57	26.76
SD15-DMD2-1Step-512(Yin et al., 2024)	26.31	25.75	25.78	26.59	26.11
SD15-15Step(Rombach et al., 2022)	26.76	26.37	26.41	27.12	26.66
SD15-25Step(Rombach et al., 2022)	27.04	26.57	26.61	27.30	26.88
SDXL-base-15step(Podell et al., 2023)	28.25	27.27	27.43	27.43	27.60
SD15-SiDLSG-1Step(report)(Zhou et al., 2024a)	27.39	26.65	26.58	27.30	26.98
SD15-SiDLSG-1Step(Zhou et al., 2024a)	26.37	25.85	25.88	26.73	26.20
PixelArt-α-25Step-512(Chen et al., 2023)	28.77	27.92	27.96	28.37	28.25
PixelArt-α-15Step-512(Chen et al., 2023)	28.68	27.85	27.87	28.29	28.17
SD15-DI++-1Step(Luo, 2024)†	28.42	27.84	28.01	28.19	28.12
SDXL-DMD2-1Step-1024(Yin et al., 2024)($\alpha_c = 8$)	28.45	27.52	27.52	27.75	27.81
SD15-DI*-1Step($\alpha_r = 1000, \alpha_c = 1.5$)(ours)	28.56	28.05	28.17	28.31	28.27
SDXL-DI*-1Step-1024($\alpha_r = 1000, \alpha_c = 8.0$)(ours)	28.74	28.04	28.14	28.12	28.26
DiT-DI*-1Step($\alpha_r = 10, \alpha_c = 4.5$)(ours)	28.78	28.31	28.48	28.37	28.48
DiT-DI*-1Step($\alpha_r = 1, \alpha_c = 4.5$)(ours)	**29.13**	**28.51**	**28.51**	**28.63**	**28.70**
SD15-15step(Rombach et al., 2022)	23.43	22.91	22.76	24.17	23.32
SDXL-base-15step(Podell et al., 2023)	29.71	27.69	27.71	25.46	27.64
PixlArt-α-15step(Chen et al., 2023)	31.31	29.86	29.56	28.49	29.81
SD15-SiDLSG-1step(Zhou et al., 2024a)	22.54	21.53	21.37	23.09	22.13
SDXL-DMD2-1step(Yin et al., 2024)	29.72	27.96	27.64	26.55	27.97
RCM-IR-2step(Li et al., 2024)	29.65	31.15	32.00	31.03	30.95
SD15-DI*-1step(ours)	29.00	28.88	29.17	27.68	28.68
SDXL-DI*-1Step-1024(ours)	30.74	30.03	30.05	28.00	29.71
DiT-DI*-1Step(ours)	**33.05**	**32.37**	**32.10**	**30.07**	**31.90**

BROADER IMPACT STATEMENT

This work is motivated by our aim to increase the positive impact of one-step text-to-image generative models toward satisfying human preferences. By default, one-step generators are either trained over large-scale image-caption pair datasets or distilled from pre-trained diffusion models, which convey only subjective knowledge without human instructions.

Our results indicate that the proposed approach is promising for making one-step generative models more aesthetic, and more preferred by human users. In the longer term, alignment failures could lead to more severe consequences, particularly if these models are deployed in safety-critical situations. For instance, if alignment failures occur, the one-step text-to-image model may generate toxic images with misleading information, and horrible images that can potentially be scary to users. We strongly recommend using our human preference alignment techniques together with AI safety checkers for text-to-image generation to prevent undesirable negative impacts.

REPRODUCIBILITY STATEMENT

We provide extensive details of experimental settings and hyperparameters to reproduce our experimental results. We plan to release our code to ensure transparency and reproducibility of the results.

REFERENCES

Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128, 2006.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

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`.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James T. Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *ArXiv*, abs/2310.00426, 2023. URL `https://api.semanticscholar.org/CorpusID:263334265`.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023.

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *ArXiv*, abs/2210.11427, 2022.

Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint arXiv:2309.15807*, 2023.

Wei Deng, Weijian Luo, Yixin Tan, Marin Biloš, Yu Chen, Yuriy Nevmyvaka, and Ricky TQ Chen. Variational schr\" odinger diffusion models. *arXiv preprint arXiv:2405.04795*, 2024.

Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. Cogview2: Faster and better text-to-image generation via hierarchical transformers. *Advances in Neural Information Processing Systems*, 35:16890–16902, 2022.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yasong Feng, Weijian Luo, Yimin Huang, and Tianyu Wang. A lipschitz bandits approach for continuous hyperparameter optimization. *arXiv preprint arXiv:2302.01539*, 2023.

Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=b6XvK2de99`.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Josh Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. *arXiv preprint arXiv:2306.05544*, 2023.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10696–10706, 2022.

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

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023a.

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*, pp. 10124–10134, 2023b.

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*, pp. 8110–8119, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

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.

Heeseung Kim, Sungwon Kim, and Sungroh Yoon. Guided-tts: A diffusion model for text-to-speech via classifier guidance. In *International Conference on Machine Learning*, pp. 11119–11133. PMLR, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *arXiv preprint arXiv:2305.01569*, 2023.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Jiachen Li, Weixi Feng, Wenhu Chen, and William Yang Wang. Reward guided latent consistency distillation. *arXiv preprint arXiv:2403.11027*, 2024.

Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL `http://arxiv.org/abs/1405.0312`. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

Xingchao Liu, Chengyue Gong, Lemeng Wu, Shujian Zhang, Hao Su, and Qiang Liu. Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. *arXiv preprint arXiv:2112.01573*, 2021.

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.

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*, 2023a.

Weijian Luo. A comprehensive survey on knowledge distillation of diffusion models. *arXiv preprint arXiv:2304.04262*, 2023.

Weijian Luo. Diff-instruct++: Training one-step text-to-image generator model to align with human preferences. *arXiv preprint arXiv:2410.18881*, 2024.

Weijian Luo and Zhihua Zhang. Data prediction denoising models: The pupil outdoes the master, 2024. URL `https://openreview.net/forum?id=wYmcfur889`.

Weijian Luo, Hao Jiang, Tianyang Hu, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Training energy-based models with diffusion contrastive divergences. *arXiv preprint arXiv:2307.01668*, 2023b.

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, 2024a.

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, 2024b.

Weijian Luo, Zemin Huang, Zhengyang Geng, J Zico Kolter, and Guo-Jun Qi. One-step diffusion distillation through score implicit matching. *arXiv preprint arXiv:2410.16794*, 2024c.

Weijian Luo, Boya Zhang, and Zhihua Zhang. Entropy-based training methods for scalable neural implicit samplers. *Advances in Neural Information Processing Systems*, 36, 2024d.

Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Thuan Hoang Nguyen and Anh Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. *arXiv preprint arXiv:2312.05239*, 2023.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.

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.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Ashwini Pokle, Zhengyang Geng, and J Zico Kolter. Deep equilibrium approaches to diffusion models. *Advances in Neural Information Processing Systems*, 35:37975–37990, 2022.

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

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

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*, pp. 8748–8763. PMLR, 2021.

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, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

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.

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*, pp. 10684–10695, 2022.

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

Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In *International conference on machine learning*, pp. 30105–30118. PMLR, 2023a.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023b.

Christoph Schuhmann. Laion-aesthetics. `https://laion.ai/blog/laion-aesthetics/`, 2022. Accessed: 2023 - 11- 10.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

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

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.

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

Yuda Song, Zehao Sun, and Xuanwu Yin. Sdxs: Real-time one-step latent diffusion models with image conditions. *arXiv preprint arXiv:2403.16627*, 2024.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Yifei Wang, Weimin Bai, Weijian Luo, Wenzheng Chen, and He Sun. Integrating amortized inference with diffusion models for learning clean distribution from corrupted images. *arXiv preprint arXiv:2407.11162*, 2024.

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. *arXiv preprint arXiv:2305.16213*, 2023.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

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

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023a.

Xingqian Xu, Zhangyang Wang, Gong Zhang, Kai Wang, and Humphrey Shi. Versatile diffusion: Text, images and variations all in one diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7754–7765, 2023b.

Shuchen Xue, Mingyang Yi, Weijian Luo, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhi-Ming Ma. SA-solver: Stochastic adams solver for fast sampling of diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=f6a9XVFYIo`.

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.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8941–8951, 2024.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *arXiv preprint arXiv:2311.18828*, 2023.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024.

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.

Boya Zhang, Weijian Luo, and Zhihua Zhang. Enhancing adversarial robustness via score-based optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL `https://openreview.net/forum?id=MOAHXRzHhm`.

Boya Zhang, Weijian Luo, and Zhihua Zhang. Purify++: Improving diffusion-purification with advanced diffusion models and control of randomness. *arXiv preprint arXiv:2310.18762*, 2023b.

Bowen Zheng and Tianming Yang. Diffusion models are innate one-step generators. *arXiv preprint arXiv:2405.20750*, 2024.

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.

Mingyuan Zhou, Zhendong Wang, Huangjie Zheng, and Hai Huang. Long and short guidance in score identity distillation for one-step text-to-image generation. *arXiv preprint arXiv:2406.01561*, 2024a.

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. *arXiv preprint arXiv:2404.04057*, 2024b.

Yufan Zhou, Ruiyi Zhang, Changyou Chen, Chunyuan Li, Chris Tensmeyer, Tong Yu, Jiuxiang Gu, Jinhui Xu, and Tong Sun. Towards language-free training for text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17907–17917, 2022.

A THEORY

A.1 PROOF OF THEOREM 3.1

Proof. Recall that $p_\theta(\cdot)$ is induced by the generator $g_\theta(\cdot)$, therefore the sample is obtained by $\boldsymbol{x}_0 = g_\theta(\boldsymbol{z}|\boldsymbol{c}), \boldsymbol{z} \sim p_z$. The term $\boldsymbol{x}$ contains parameter through $\boldsymbol{x}_0 = g_\theta(\boldsymbol{z}|\boldsymbol{c}), \boldsymbol{z} \sim p_z$. To demonstrate the parameter dependence, we use the notation $p_\theta(\cdot)$. Note that $p_{ref}(\cdot)$ is the reference distribution. The alignment objective writes

$$\mathcal{L}_{Orig}(\theta) = \mathbb{E}_{\substack{\boldsymbol{z}\sim p_z, \\ \boldsymbol{x}_0 = g_\theta(\boldsymbol{z}|\boldsymbol{c})}} \big[-\alpha r(\boldsymbol{x}_0, \boldsymbol{c}) \big] + \mathbf{D}^{[0,T]}(p_\theta, p_{ref}) \tag{A.1}$$

The first loss term of (A.1) $\alpha r(\boldsymbol{x}_0, \boldsymbol{c})$ is easy to compute by directly pushing the generated sample $\boldsymbol{x}_0$ and the text prompt $\boldsymbol{c}$ into the reward model $r(\cdot,\cdot)$. However, the second loss term (A.2) is intractable because we do not explicitly know the relation between θ and $p_{\theta,t}(\cdot)$.

$$\mathbf{D}^{[0,T]}(p_\theta, p_{ref}) \coloneqq \int_{t=0}^{T} w(t) \mathbb{E}_{\boldsymbol{x}_t \sim \pi_t} \Big\{ \mathbf{d}(\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)) \Big\} \mathrm{d}t, \tag{A.2}$$

We turn to derive the equivalent loss for $\mathbf{D}^{[0,T]}(p_\theta, p_{ref})$. First we take the θ gradient of (A.2), show

$$\frac{\partial}{\partial \theta} \mathbf{D}^{[0,T]}(p_\theta, p_{ref}) = \frac{\partial}{\partial \theta} \int_{t=0}^{T} w(t) \mathbb{E}_{\boldsymbol{x}_t \sim \pi_t} \Big\{ \mathbf{d}(\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)) \Big\} \mathrm{d}t \tag{A.3}$$

$$= \mathbb{E}_{t, \boldsymbol{x}_t \sim \pi_t} w(t) \Big\{ \mathbf{d}'(\boldsymbol{y}_t) \Big\}^T \frac{\partial}{\partial \theta} \boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) \tag{A.4}$$

Notice that $p_{\theta,t}(\cdot)$ is induced by first generating samples with one-step generator then adding noise with diffusion process (2.1), we do not know the term $\frac{\partial}{\partial \theta} \boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)$. Therefore the gradient formula (A.4) is intractable. However, we will show that a tractable loss function can recover the intractable gradient (A.4), and therefore can be used for minimizing (A.2). Our proof is inspired by the theory from Vincent (2011), Zhou et al. (2024b) and Luo et al. (2024c).

We first present a so-called Score-projection identity (Theorem A.1), which has been studied in Zhou et al. (2024b) and Vincent (2011):

Theorem A.1. Let $\boldsymbol{u}(\cdot)$ be a θ-free vector-valued function under mild conditions, the identity holds:

$$\mathbb{E}_{\substack{\boldsymbol{x}_0 \sim p_{\theta,0}, \\ \boldsymbol{x}_t | \boldsymbol{x}_0 \sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} \boldsymbol{u}(\boldsymbol{x}_t)^T \Big\{ \boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \Big\} = 0, \quad \forall \theta. \tag{A.5}$$

We give a short proof of Theorem A.1 as a clarification. Readers can also refer to Vincent (2011) or Zhou et al. (2024b) as a reference.

Recall the relation between $\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)$ and $\nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$, we know

$$\begin{aligned}
\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) &= \nabla_{\boldsymbol{x}_t} \log \int p_{\theta,0}(\boldsymbol{x}_0) q_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \mathrm{d}\boldsymbol{x}_0 \\
&= \frac{\int p_{\theta,t}(\boldsymbol{x}_0) \nabla_{\boldsymbol{x}_t} q_t(\boldsymbol{x}_t|\boldsymbol{x}_0) \mathrm{d}\boldsymbol{x}_0}{p_{\theta,t}(\boldsymbol{x}_t)} \\
&= \int \frac{\nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0) p_{\theta,t}(\boldsymbol{x}_0) q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}{p_{\theta,t}(\boldsymbol{x}_t)} \mathrm{d}\boldsymbol{x}_0
\end{aligned}$$

We have

$$\begin{aligned}
&\mathbb{E}_{\substack{\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} \boldsymbol{u}(\boldsymbol{x}_t)^T \boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{x}_t\sim p_{\theta,t}} \boldsymbol{u}(\boldsymbol{x}_t)^T \boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) \\
&= \int p_{\theta,t}(\boldsymbol{x}_t)\boldsymbol{u}(\boldsymbol{x}_t)^T \boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\mathrm{d}\boldsymbol{x}_t \\
&= \int p_{\theta,t}(\boldsymbol{x}_t)\boldsymbol{u}(\boldsymbol{x}_t)^T \int \frac{\nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_{\theta,t}(\boldsymbol{x}_0)q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}{p_{\theta,t}(\boldsymbol{x}_t)}\mathrm{d}\boldsymbol{x}_0\mathrm{d}\boldsymbol{x}_t \\
&= \int \boldsymbol{u}(\boldsymbol{x}_t)^T \int \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_{\theta,t}(\boldsymbol{x}_0)q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_0\mathrm{d}\boldsymbol{x}_t \\
&= \int\int \boldsymbol{u}(\boldsymbol{x}_t)^T \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)p_{\theta,t}(\boldsymbol{x}_0)q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_0\mathrm{d}\boldsymbol{x}_t \\
&= \mathbb{E}_{\substack{\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} \boldsymbol{u}(\boldsymbol{x}_t)^T \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)
\end{aligned}$$

If we take the θ gradient on both sides of (A.5), we have

$$\begin{aligned}
0 = \mathbb{E}_{\substack{\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} &\left\{\frac{\partial}{\partial \boldsymbol{x}_t}\left[\boldsymbol{u}(\boldsymbol{x}_t)^T\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\}\right]\frac{\partial \boldsymbol{x}_t}{\partial\theta}\right. \\
&\left. - \boldsymbol{u}(\boldsymbol{x}_t)^T\frac{\partial}{\partial \boldsymbol{x}_0}\left[\nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right]\frac{\partial \boldsymbol{x}_0}{\partial\theta}\right\} + \mathbb{E}_{\boldsymbol{x}_t\sim p_{\theta,t}}\boldsymbol{u}(\boldsymbol{x}_t)^T\frac{\partial}{\partial\theta}\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\right\}
\end{aligned} \tag{A.6}$$

So we have an identity

$$\mathbb{E}_{\boldsymbol{x}_t\sim p_{\theta,t}}\boldsymbol{u}(\boldsymbol{x}_t)^T\frac{\partial}{\partial\theta}\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\right\} = -\frac{\partial}{\partial\theta}\mathbb{E}_{\substack{\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\left\{\boldsymbol{u}(\boldsymbol{x}_t)\left\{\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\}\right\}$$

Notice that the left-hand side of equation (A.7) can be interpreted as the gradient of the loss function when the parameter dependency of the sampling distribution is cut off, i.e.

$$\mathbb{E}_{\boldsymbol{x}_t\sim p_{\theta,t}}\boldsymbol{u}(\boldsymbol{x}_t)^T\frac{\partial}{\partial\theta}\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\right\} = \frac{\partial}{\partial\theta}\mathbb{E}_{\boldsymbol{x}_t\sim p_{\mathrm{sg}[\theta],t}}\left\{\boldsymbol{u}(\boldsymbol{x}_t)^T\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\right\} \tag{A.7}$$

Therefore we have the final equation

$$\frac{\partial}{\partial\theta}\mathbb{E}_{\boldsymbol{x}_t\sim p_{\mathrm{sg}[\theta],t}}\left\{\boldsymbol{u}(\boldsymbol{x}_t)^T\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\right\} = -\frac{\partial}{\partial\theta}\mathbb{E}_{\substack{\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\left\{\boldsymbol{u}(\boldsymbol{x}_t)\left\{\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\}\right\} \tag{A.8}$$

which holds for arbitrary function $\boldsymbol{u}(\cdot)$ and parameter θ. If we set

$$\begin{aligned}
\boldsymbol{u}(\boldsymbol{x}_t) &= \mathbf{d}'(\boldsymbol{y}_t) \\
\boldsymbol{y}_t &= \boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)
\end{aligned}$$

Then we formally have

$$\begin{aligned}
&\frac{\partial}{\partial\theta}\mathbb{E}_{\boldsymbol{x}_t\sim p_{\mathrm{sg}[\theta],t}}\left\{\mathbf{d}'(\boldsymbol{y}_t)\right\}^T\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t)\right\} \\
&= \frac{\partial}{\partial\theta}\mathbb{E}_{\substack{\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}}\left\{-\mathbf{d}'(\boldsymbol{y}_t)\right\}^T\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\}
\end{aligned} \tag{A.9}$$

This means that we can use the θ gradient of a tractable loss:

$$\mathbb{E}_{\substack{t,\boldsymbol{x}_0\sim p_{\theta,0},\\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} w(t)\left\{-\mathbf{d}'(\boldsymbol{y}_t)\right\}^T\left\{\boldsymbol{s}_{p_{\theta,t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\} \tag{A.10}$$

to replace the wanted θ gradient (A.3), which can minimize the regularization loss (A.2).

Combining $r(\boldsymbol{x}_0, \boldsymbol{c})$ and (A.10), we have the practical loss

$$\begin{aligned}
\mathcal{L}_{DI*}(\theta) =& \mathbb{E}_{\substack{\boldsymbol{z}\sim p_z,\\ \boldsymbol{x}_0=g_\theta(\boldsymbol{z})}}\Bigg[-\alpha r(\boldsymbol{x}_0,\boldsymbol{c}) \\
&+ \mathbb{E}_{\substack{t,\boldsymbol{x}_t|\boldsymbol{x}_0\\ \sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)}} w(t)\left\{-\mathbf{d}'(\boldsymbol{y}_t)\right\}^T\left\{\boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t}\log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\}\mathrm{d}t\Bigg]
\end{aligned} \tag{A.11}$$

□

Remark A.2. In practice, most commonly used forward diffusion processes can be expressed as a form of scale and noise addition:

$$\boldsymbol{x}_t = \alpha(t)\boldsymbol{x}_0 + \beta(t)\epsilon, \quad \epsilon \sim \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{I}). \tag{A.12}$$

So the term $\boldsymbol{x}_t$ in equation (A.11) can be instantiated as $\boldsymbol{z} \sim p_z,\ \epsilon \sim \mathcal{N}(\epsilon; \mathbf{0}, \boldsymbol{I}),\ \boldsymbol{x}_t = \alpha(t)\boldsymbol{x}_0 + \beta(t)\epsilon$.

A.2 Proof of Theorem 3.2

Proof. Recall the definition of the classifier-free reward (3.9). The negative reward writes

$$-r(\boldsymbol{x}_0, \boldsymbol{c}) = -\mathbb{E}_{t,\boldsymbol{x}_t\sim p_{\theta,t}} w(t) \log \frac{p_{ref}(\boldsymbol{x}_t|t, \boldsymbol{c})}{p_{ref}(\boldsymbol{x}_t|t)}$$

This reward will put a higher reward on those samples that have higher class-conditional probability than unconditional probability, therefore encouraging class-conditional sampling. It is clear that

$$\begin{aligned}\frac{\partial}{\partial\theta}\Big\{-r(\boldsymbol{x}_0, \boldsymbol{c})\Big\} &= -\mathbb{E}_{t,\boldsymbol{x}_t\sim p_{\theta,t}} w(t)\Big\{\nabla_{\boldsymbol{x}_t}\log p_{ref}(\boldsymbol{x}_t|t, \boldsymbol{c}) - \nabla_{\boldsymbol{x}_t}\log p_{ref}(\boldsymbol{x}_t|t)\Big\}\frac{\partial \boldsymbol{x}_t}{\partial\theta} \\ &= -\mathbb{E}_{t,\boldsymbol{x}_t\sim p_{\theta,t}} w(t)\Big\{\boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t, \boldsymbol{c}) - \boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t, \boldsymbol{\varnothing})\Big\}\frac{\partial \boldsymbol{x}_t}{\partial\theta}\end{aligned} \tag{A.13}$$

Therefore, we can see that the equivalent loss

$$\mathcal{L}_{cfg}(\theta) = \mathbb{E}_{\substack{t,\boldsymbol{z}\sim p_z,\boldsymbol{x}_0=g_\theta(\boldsymbol{z}|\boldsymbol{c}) \\ \boldsymbol{x}_t|\boldsymbol{x}_0\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}} w(t)\Big\{\boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t, \boldsymbol{c}) - \boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t, \boldsymbol{\varnothing})\Big\}^T \boldsymbol{x}_t \tag{A.14}$$

recovers the gradient formula (A.13). □

B Important Materials for Main Content

B.1 Prompts for Figure 1 and Figure 2

Prompts for Figure 1 (from upper left to bottom right):

- *A girl examining an ammonite fossil*;
- *A squirrel driving a toy car*;
- *A portrait of a statue of the Egyptian god Anubis wearing aviator goggles, white t-shirt and leather jacket. The city of Los Angeles is in the background*;
- *A still image of a humanoid cat posing with a hat and jacket in a bar*;
- *A photograph of the inside of a subway train. There are red pandas sitting on the seats. One of them is reading a newspaper. The window shows the jungle in the background*;
- *A capybara made of voxels sitting in a field*;
- *A teddy bear on a skateboard in times square*;
- *A sloth in a go kart on a race track. The sloth is holding a banana in one hand. There is a banana peel on the track in the background*;
- *A close-up photo of a wombat wearing a red backpack and raising both arms in the air*;
- *A small cactus with a happy face in the Sahara desert*;
- *Baker proudly displays her white dog cake in her kitchen*;
- *A bowl with rice, broccoli and a purple relish*;
- *An inlet filled with boats of all kinds*;
- *A black cat sitting on top of the hood of a car*;
- *A woman wearing a cowboy hat face to face with a horse*.

Prompts for Figure 2. The prompts are listed from the up rows to the bottom row:

- *Pirate ship sailing into a bioluminescence sea with a galaxy in the sky, epic, 4k, ultra*;
- *Digital 2D, Miyazaki's style, ultimate detailed, tiny finnest details, futuristic, sci-fi, magical dreamy landscape scenery, small cute girl living alone with plushified friendly big tanuki in the gigantism of wilderness, intricate round futuristic simple multilayered architecture, habitation cabin in the trees, dramatic soft lightning, rule of thirds, cinematic*;
- *saharian landscape at sunset, 4k ultra realism, BY Anton Gorlin, trending on artstation, sharp focus, studio photo, intricate details, highly detailed, by greg rutkowski*.

B.2 Meanings of Hyper-parameters.

Meanings of Hyper-parameters. As in Algorithm 1, the overall algorithms consist of two alternative updating steps. The first step is to update ψ of the assistant diffusion model by fine-tuning it with student-generated data. Therefore the assistant diffusion $\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})$ can approximate the score function of student generator distribution. This step means that the assistant diffusion needs to communicate with the student to know the student's status. The second step updates the generator by minimizing the tractable loss (3.7) using SGD-based optimization algorithms such as Adam (Kingma & Ba, 2014). This step means that the teacher and the assistant diffusion discuss and incorporate the student's interests to instruct the student generator.

As we can see in Algorithm 1 (as well as Algorithm 2). Each hyperparameter has its intuitive meaning. The reward scale parameter α_{rew} controls the strength of human preference alignment. The larger the α_{rew} is, the stronger the generator is aligned with human preferences. However, the drawback for a too large α_{rew} might be the loss of diversity and reality. Besides, we empirically find that larger α_{rew} leads to richer generation details and better generation layouts. But a very large α_{rew} results in unrealistic and painting-like images.

The CFG reward scale controls the strength of using CFG rewards when training. We empirically find that the best CFG scale for Diff-Instruct* is the same as the best CFG scale for sampling from the reference diffusion model. However, α_{cfg} may conflicts with α_{rew}. In the Stable Diffusion 1.5 experiment, we find that using a large CFG reward scale leads to worse human preferences. Therefore, the proper combination of $(\alpha_{rew}, \alpha_{cfg})$ asks for careful tuning.

The diffusion model weighting $\lambda(t)$ and the generator loss weighting $w(t)$ controls the strengths put on each time level of updating assistant diffusion and the student generator. We empirically find that it is decent to set $\lambda(t)$ to be the same as the default training weighting function for the reference diffusion. And it is decent to set the $w(t) = 1$ for all time-levels in practice. In the following section, we give more discussions on Diff-Instruct*.

B.3 More Discussions on Diff-Instruct*

Flexible Choices of Divergences. Clearly, various choices of distance function $\mathbf{d}(.)$ result in different training algorithms. In this part, we discuss two instances. The first choice distance function is a simple squared distance, i.e. $\mathbf{d}(\boldsymbol{y}_t) = \|\boldsymbol{y}_t\|_2^2$. The corresponding derivative term writes $\mathbf{d}'(\boldsymbol{y}_t) = 2\boldsymbol{y}_t$. In fact, such a distance function recovers the practical diffusion distillation loss studied in Zhou et al. (2024b;a). The second distance is the pseudo-Huber distance, which shows more robust performances than the simple squared distance. The pseudo-Huber distance is defined with $\boldsymbol{d}(\boldsymbol{y}) := \sqrt{\|\boldsymbol{y}_t\|_2^2 + c^2} - c$, where c is a pre-defined positive constant. The corresponding regularization loss (3.6) writes

$$\mathbf{D}^{[0,T]}(p_\theta, p_{ref}) = -\left\{\frac{\boldsymbol{y}_t}{\sqrt{\|\boldsymbol{y}_t\|_2^2 + c^2}}\right\}^T \left\{\boldsymbol{s}_\psi(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log q_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\}. \tag{B.1}$$

Here $\boldsymbol{y}_t := \boldsymbol{s}_{p_{\mathrm{sg}[\theta],t}}(\boldsymbol{x}_t) - \boldsymbol{s}_{q_t}(\boldsymbol{x}_t)$.

DI* Does Not Need Image Data When Training. One appealing advantage of DI* is the image data-free property, which means that DI* requires neither the image datasets nor synthetic images that are generated by reference diffusion models. This advantage distinguishes DI* from previous fine-tuning methods such as generative adversarial training (Goodfellow et al., 2014) which require training additional neural classifiers over image data, as well as those fine-tuning methods over large-scale synthetic or curated datasets.

Table 3: HPSv2.0 score of SDXL-based 1-step model alignment using Diff-Instruct*. We mark the increment over the initial model with green number to demonstrate its trend. We can see that the increment converges to +0.42.

K IMAGES	0	102	205	307	410	512	614	717	819	922
CONCEPT-ART↑	27.52	27.64	27.64	27.70	27.78	27.90	27.98	28.02	28.04	28.05
PHOTO↑	27.75	27.85	27.83	27.88	27.98	28.05	28.10	28.09	28.12	28.09
ANIMATION↑	28.45	28.52	28.58	28.56	28.68	28.76	28.79	28.78	28.74	28.73
PAINTING↑	27.52	27.60	27.68	27.73	27.85	27.95	28.04	28.09	28.14	28.11
AVG HPSV2.0↑	27.81	27.90(+0.09)	27.93(+0.11)	27.97(+0.17)	28.07(+0.26)	28.17(+0.36)	28.23(+0.42)	28.24(+0.43)	28.26(+0.05)	28.24(+0.43)

The Choice of Generator is Flexible across Broader Applications. Another interesting property of DI* for alignment is its wide flexibility in the choice of generator models. We can see that, the theory of DI* only requires the generator to be able to generate output images (or data of other modalities) that are differentiable with the generator's parameters. This makes DI* a universal training method for two reasons. 1) the choice of generator architecture is flexible. The network architectures for diffusion models require the input and output to have the same dimensions. However, the DI* does not assign such a restriction to generator network choices. Therefore, pre-trained GAN generators, such as StyleGAN-T (Sauer et al., 2023a) and GigaGAN (Kang et al., 2023b) are also compatible with DI*. Besides, we have also shown in Section 5.2 that DI* is compatible with both UNet-based and DiT-based generator architectures. 2) student networks in broader applications may also satisfy the requirements of DI*. For instance, the neural radiance field (Mildenhall et al., 2021) model used in text-to-3D generation using text-to-2D diffusion models can also be viewed as a generator. Therefore DI* can be used for such scenarios to incorporate human preference in the training process. Readers can read Poole et al. (2022), Wang et al. (2023) for more introductions.

B.4 EXPERIMENT DETAILS FOR PRE-TRAINING AND ALIGNMENT

The human preference score (HPSv2.0) trend of DI*-SDXL-1step model. During the training of the DI*-SDXL-1step model, we monitor the change of the HPSv2.0 score as an out-of-sample validation metric. We initialize the 1-step model with DMD2-SDXL-1step model (Yin et al., 2024), which is a pretty solid SDXL-based one-step diffusion distillation model. We set the learning rate of both the one-step generator and the online diffusion model to be $1e-5$ and set an exponential moving average decay rate of 0.9 for faster convergence. Following the same setting as SD1.5 and PixelArt-α experiment, we use the ImageReward as the explicit reward, while using a CFG scale of 8.0 for implicit CFG reward. Table 3 records the HPSv2.0 trend of the alignment process using Diff-Instruct*.

Detailed Experiment Settings for SD1.5 Experiment. For experiments of SD1.5, we use the open-sourced SD1.5 of a resolution of 512×512 as our reference diffusion in Algorithm 1. We implement our experiments based on SiD-LSG (Zhou et al., 2024a) [1], which provides a high-quality codebase for diffusion model training and distillation. We construct the one-step generator with the same architecture as the reference SD1.5 model, following the same configuration of SiD-LSG. We use the prompts of the LAION-AESTHETIC dataset with an aesthetic score larger than 6.25, which resulting a total of 3M text prompts. We only prepare the text prompts since DI* does not need image datasets. We use the off-the-shelf Image Reward [2] as our explicit reward model. To better explore the advantages of our score-based divergence over traditional KL divergence, we refer to a recent work (Luo, 2024) that uses KL divergence for training and conducts a detailed comparison between score-based divergences that DI* uses and KL divergences in previous works.

Detailed Experiment Settings for PixelArt-α Experiment. Different from Stabld Diffusion models, the PixArt-α model is a high-quality open-sourced text-to-image diffusion model. It uses a diffusion transformer (Peebles & Xie, 2022) to learn marginal score functions in a latent space encoded by a down-sampled variational auto-encoder (VAE) (Rombach et al., 2022). For the text conditioning mechanism, the PixelArt-α model uses a T5-XXL text encoder(Raffel et al., 2020; Tay et al., 2021), which makes the model able to understand long prompts without an obvious length

[1] https://github.com/mingyuanzhou/SiD-LSG
[2] https://github.com/THUDM/ImageReward

restriction. We use the DiT architecture and the 0.6B PixArt-α of a resolution of 512×512 as our reference diffusion to demonstrate the compatibility of DI* for different kinds of neural network architectures. We use the prompts from the SAM-LLaVA-Caption-10M dataset as our prompt dataset. The SAM-LLaVA-Caption-10M dataset contains the images collected by Kirillov et al. (2023), together with text descriptions that are captioned by LLaVA model (Liu et al., 2024). The SAM-LLaVA-Caption-10M dataset is used for training the PixelArt-α model. Since the PixelArt-α diffusion model uses a T5-XXL, which is memory and computationally expensive. To speed up the alignment training, we pre-encoded the text prompts using the T5-XXL text encoder, saved the encoded embedding vectors, and built the data loaders in-house.

We follow the setting of Diff-Instruct (Luo et al., 2024a) to use the same neural network architecture as the reference diffusion model for the one-step generator. The PixelArt-α model is trained using so-called VP diffusion(Song et al., 2020), which first scales the data in the latent space, then adds noise to the scaled latent data. We reformulate the VP diffusion as the form of so-called *data-prediction* proposed in EDM paper (Karras et al., 2022) by re-scaling the noisy data with the inverse of the scale that has been applied to data with VP diffusion. Under the data-prediction formulation, we select a fixed noise σ_{init} level to be $\sigma_{init} = 2.5$ following the Diff-Instruct and SiD (Zhou et al., 2024b). For generation, we first generate a Gaussian vector $\boldsymbol{z} \sim p_z = \mathcal{N}(\mathbf{0}, \sigma_{init}^2 \mathbf{I})$. Then we input $\boldsymbol{z}$ into the generator to generate the latent. The latent vector can then be decoded by the VAE decoder to turn into an image if needed.

We put the details of how to construct the one-step generator in the following paragraphs. We also initialize the assistant diffusion model with the same weight as the reference diffusion. We use the Image Reward as the human preference reward and use the Diff-Instruct* algorithm 1 (or equivalently the algorithm 2) to train the generator. We also used the Adam optimizer with the parameter $(\beta_1, \beta_2) = (0.0, 0.999)$ for both the generator and the assistant diffusion with a batch size of 128, implemented with BF16 numerical format and the accumulate-gradient training technique. We use a fixed exponential moving average decay (EMA) rate of 0.95 for all training trials. After the training, the generator aligned with both strong CFG and reward model shows significantly improved aesthetic appearance, better generation layout, and richer image details. Figure 2 shows a demonstration of the generated images using our aligned one-step generator with a CFG scale α_{cfg} of 4.5 and a reward scale α_{rew} of 10.0.

Construction of the one-step generator. We follow the experiment setting of Diff-Instruct (Luo et al., 2024b), generalizing its CIFAR10 experiment to text-to-image generation. Notice that the Diff-Instruct uses the EDM model (Karras et al., 2022) to formulate the diffusion model, as well as the one-step generator. We start with a brief introduction to the EDM model.

The EDM model depends on the diffusion process

$$\mathrm{d}\boldsymbol{x}_t = t\mathrm{d}\boldsymbol{w}_t, t \in [0, T]. \tag{B.2}$$

Samples from the forward process (B.2) can be generated by adding random noise to the output of the generator function, i.e., $\boldsymbol{x}_t = \boldsymbol{x}_0 + t\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a Gaussian vector. The EDM model also reformulates the diffusion model's score matching objective as a denoising regression objective, which writes,

$$\mathcal{L}(\psi) = \int_{t=0}^{T} \lambda(t) \mathbb{E}_{\boldsymbol{x}_0 \sim p_0, \boldsymbol{x}_t | \boldsymbol{x}_0 \sim p_t(\boldsymbol{x}_t | \boldsymbol{x}_0)} \|\boldsymbol{d}_\psi(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\|_2^2 \mathrm{d}t. \tag{B.3}$$

Where $\boldsymbol{d}_\psi(\cdot)$ is a denoiser network that tries to predict the clean sample by taking noisy samples as inputs. Minimizing the loss (B.3) leads to a trained denoiser, which has a simple relation to the marginal score functions as:

$$\boldsymbol{s}_\psi(\boldsymbol{x}_t, t) = \frac{\boldsymbol{d}_\psi(\boldsymbol{x}_t, t) - \boldsymbol{x}_t}{t^2} \tag{B.4}$$

Under such a formulation, we actually have pre-trained denoiser models for experiments. Therefore, we use the EDM notations in later parts.

Let $\boldsymbol{d}_\theta(\cdot)$ be pretrained EDM denoiser models. Owing to the denoiser formulation of the EDM model, we construct the generator to have the same architecture as the pre-trained EDM denoiser with a pre-selected index t^*, which writes

$$\boldsymbol{x}_0 = g_\theta(\boldsymbol{z}) \coloneqq \boldsymbol{d}(\boldsymbol{z}, t^*), \;\; \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, (t^*)^2 \mathbf{I}). \tag{B.5}$$

We initialize the generator with the same parameter as the teacher EDM denoiser model.

Time index distribution. When training both the EDM diffusion model and the generator, we need to randomly select a time t in order to approximate the integral of the loss function (B.3). The EDM model has a default choice of t distribution as log-normal when training the diffusion (denoiser) model, i.e.

$$t \sim p_{EDM}(t): \quad t = \exp(s) \tag{B.6}$$

$$s \sim \mathcal{N}(P_{mean}, P_{std}^2), \quad P_{mean} = -2.0, P_{std} = 2.0. \tag{B.7}$$

And a weighting function

$$\lambda_{EDM}(t) = \frac{(t^2 + \sigma_{data}^2)}{(t \times \sigma_{data})^2}. \tag{B.8}$$

In our algorithm, we follow the same setting as the EDM model when updating the online diffusion (denoiser) model.

Weighting function. For the assistant diffusion updates in both pre-training and alignment, we use the same $\lambda_{EDM}(t)$ (B.8) weighting function as EDM when updating the denoiser model. When updating the generator, we use a specially designed weighting function, which writes:

$$w_{Gen}(t) = \frac{1}{\|\boldsymbol{d}_{\psi}(\mathrm{sg}[\boldsymbol{x}_t], t) - \boldsymbol{d}_{q_t}(\mathrm{sg}[\boldsymbol{x}_t], t)\|_2} \tag{B.9}$$

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + t\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{B.10}$$

The notation $\mathrm{sg}[\cdot]$ means stop-gradient of the parameter. Such a weighting function helps to stabilize the training.

In the both Stable Diffusion 1.5 and the PixArt-α experiments, we rewrite the PixelArt-α model in EDM formulation:

$$D_\theta(\mathbf{x}; \sigma) = \mathbf{x} - \sigma F_\theta \tag{B.11}$$

Here, following the iDDPM+DDIM preconditioning in EDM, PixelArt-α is denoted by F_θ, $\mathbf{x}$ is the image data plus noise with a standard deviation of σ, for the remaining parameters such as C_1 and C_2, we kept them unchanged to match those defined in EDM. Unlike the original model, we only retained the image channels for the output of this model. Since we employed the preconditioning of iDDPM+DDIM in the EDM, each σ value is rounded to the nearest 1000 bins after being passed into the model. For the actual values used in PixelArt-α, beta_start is set to 0.0001, and beta_end is set to 0.02. Therefore, according to the formulation of EDM, the range of our noise distribution is [0.01, 156.6155], which will be used to truncate our sampled σ. For our one-step generator, it is formulated as:

$$g_\theta(\mathbf{x}; \sigma_{\text{init}}) = \mathbf{x} - \sigma_{\text{init}} F_\theta \tag{B.12}$$

Here following Diff-Instruct to use $\sigma_{\text{init}} = 2.5$ and $\mathbf{x} \sim \mathcal{N}(0, \sigma_{\text{init}}\mathbf{I})$, we observed in practice that larger values of σ_{init} lead to faster convergence of the model, but the difference in convergence speed is negligible for the complete model training process and has minimal impact on the final results.

Detailed Quantitative Evaluations Metrics To quantitatively evaluate the performances of the generators trained with different alignment settings, we compare our generators elaborately with other open-sourced models that are either based on SD1.5 diffusion models or larger models such as SDXL. For all models, we compute four standard scores: the Image Reward (Xu et al., 2023a), the Aesthetic Score (Schuhmann, 2022), the PickScore(Kirstain et al., 2023), and the CLIP score(Radford et al., 2021). Since most existing literature tests the human preference scores with different prompts which are possibly not available, to make a fair comparison, in our experiment, we fix 1k prompts from the COCO-2017 (Lin et al., 2014) validation dataset and intensively evaluate a wide range of existing open-sourced models as SiD-LSG (Zhou et al., 2024a), SDXL-Turbo (Sauer et al., 2023b), Latent Consistency Model (LCM) (Luo et al., 2023a), Hyper-SD (Ren et al., 2024), SDXL-Lightning (Lin et al., 2024), Trajectory Consistency Model (TCD) (Zheng et al., 2024), PeReflow (Yan et al., 2024), InstaFLow(Liu et al., 2023), Diffusion DPO(Wallace et al., 2024), etc. All models are tested with the same prompts and the same computing devices.

Besides the COCO prompts, we also evaluate generators with open-sourced models with Human Preference Score v2.0 (HPSv2.0) (Wu et al., 2023) over their benchmark prompts. The HPS is a

widely used standard benchmark that evaluates models' capability of generating images of 4 styles: Animation, concept art, Painting, and Photo. The score reflects the prompt following and the human preference strength of text-to-image models. We use the HPSv2's [3] default protocols for evaluations. Since our generators based on PixArt-α are trained with SAM-recaptioned datasets, we also evaluate the performances on 30K prompts from the SAM-recap dataset.

Sample Prompts from COCO validation dataests. In this paragraph, we propose some sample prompts from the COCO validation dataset that we use for calculating scores like Image Reward and the Aesthetic Scores.

```
This wire metal rack holds several pairs of shoes and sandals
A motorcycle parked in a parking space next to another motorcycle.
A picture of a dog laying on the ground.
A loft bed with a dresser underneath it.
Two giraffes in a room with people looking at them.
A woman stands in the dining area at the table.
Birds perch on a bunch of twigs in the winter.
A small kitchen with low a ceiling
A group of baseball players is crowded at the mound.
This table is filled with a variety of different dishes.
A toy dinosaur standing on a sink next to a running faucet.
a man standing holding a game controller and two people sitting
There is a small bus with several people standing next to it.
A bottle on wine next to a glass of wine.
A big burly grizzly bear is show with grass in the background.
A man standing in front of a microwave next to pots and pans.
Three men in military suits are sitting on a bench,
Two people standing in a kitchen looking around.
A group of men playing a game of baseball on top of a baseball field.
A traffic light over a street surrounded by tall buildings.
The snowboarder has jumped high into the air from a snow ramp.
A smart phone with an image of a person on it's screen.
A man talking on his phone in the public.
A cheesy pizza sitting on top of a table.
A dog sitting on the inside of a white boat.
A guy jumping with a tennis racket in his hand.
a close up of a child next to a cake with balloons
A man holding a camera up over his left shoulder.
A plane flies over water with two islands nearby.
A young boy getting ready to catch a baseball in a grass field.
```

Listing 1: Example Prompts from COCO validation dataset.

B.5 More Discussions on Findings of Qualitative Evaluations

There are some other interesting findings when qualitatively evaluate different models.

- First, we find that the images generated by the aligned model show a better composition when organizing the contents presented in the image. For instance, the main objects of the generated image are smaller and show a more natural layout than other models, with the objects and the background iterating aesthetically. This in turn reveals human preference: human beings would prefer that the object of an image does not take up all spaces of an image;
- Second, we find that the aligned model has richer details than the unaligned model. The stronger we align the model, the richer details the model will generate. Sometimes these rich details come as a hint to the readers about the input prompts. Sometimes they just come to improve the aesthetic performance. We think this phenomenon may be caused by the fact that human prefers images with rich details. Another finding is that as the reward scale for alignment becomes stronger, the generated image from the alignment model becomes more colorful and more similar to paintings. Sometimes this leads to a loss of reality to some

[3] https://github.com/tgxs002/HPSv2

Before human-preference alignment

After human-preference alignment with DI*

Figure 4: A visual comparison of the 0.68B SD1.5-DI* model before (initialized with SiD-LSG pre-trained weights) and after alignments with DI*. We put the prompt in Appendix B.1.

Algorithm 2: Diff-Instruct* Pseudo Code.

Input: prompt dataset $\mathcal{C}$, generator $g_\theta(\boldsymbol{x}_0|\boldsymbol{z},\boldsymbol{c})$, prior distribution p_z, reward model $r(\boldsymbol{x},\boldsymbol{c})$, reward model scale α_{rew}, CFG reward scale α_{cfg}, reference diffusion model $\boldsymbol{s}_{ref}(\boldsymbol{x}_t|c,\boldsymbol{c})$, assistant diffusion $\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})$, forward diffusion $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ (2.1), assistant diffusion updates rounds K_{TA}, time distribution $\pi(t)$, diffusion model weighting $\lambda(t)$, generator IKL loss weighting $w(t)$.

while *not converge* **do**

freeze θ, update ψ for K_{TA} rounds by

1. sample prompt $\boldsymbol{c}\sim\mathcal{C}$; sample time $t\sim\pi(t)$; sample $\boldsymbol{z}\sim p_z(\boldsymbol{z})$;
2. generate fake data: $\boldsymbol{x}_0=\mathrm{sg}[g_\theta(\boldsymbol{z},\boldsymbol{c})]$; sample noisy data: $\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$;
3. update ψ by minimizing loss: $\mathcal{L}(\psi)=\lambda(t)\|\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})-\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\|_2^2$;

freeze ψ, update θ using SGD:

1. sample prompt $\boldsymbol{c}\sim\mathcal{C}$; sample time $t\sim\pi(t)$; sample $\boldsymbol{z}\sim p_z(\boldsymbol{z})$;
2. generate fake data: $\boldsymbol{x}_0=g_\theta(\boldsymbol{z},\boldsymbol{c})$; sample noisy data: $\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$;
3. explicit reward: $\mathcal{L}_{rew}(\theta)=-\alpha_{rew}r(\boldsymbol{x}_0,\boldsymbol{c})$;
4. CFG reward: $\mathcal{L}_{cfg}(\theta)=\alpha_{cfg}\cdot w(t)\big\{\boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t,\boldsymbol{c})-\boldsymbol{s}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t,\varnothing)\big\}^T\boldsymbol{x}_t$;
5. score-regularization: $\mathcal{L}_{reg}(\theta)=-w(t)\big\{\mathbf{d}'(\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})-\boldsymbol{s}_{ref}(\boldsymbol{x}_t|t,\boldsymbol{c}))\big\}^T\big\{\boldsymbol{s}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})-\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)\big\}$;
6. update θ by minimizing DI* loss: $\mathcal{L}_{DI*}(\theta)=\mathcal{L}_{rew}(\theta)+\mathcal{L}_{cfg}(\theta)+\mathcal{L}_{reg}(\theta)$;

end

return θ,ψ.

---

degree. Therefore, we think that users should choose different aligned one-step models with a trade-off between aesthetic performance and image reality according to the use case.

Figure 5: Bad generation cases by aligned DiT-DI* one-step generator model (4.5 CFG + 10.0 reward).

Table 4: Hyperparameters used for Diff-Instruct* on SD1.5 and PixArt-α experiments.

Hyperparameter	SD1.5 Experiment		PixArt-α Experiment	
	DM $\boldsymbol{s}_\psi$	Generator g_θ	DM $\boldsymbol{s}_\psi$	Generator g_θ
Learning rate	1e-5	1e-5	2e-6	2e-6
Batch size	512	512	256	256
$\sigma(t^*)$	2.5	2.5	2.5	2.5
$Adam\ \beta_0$	0.0	0.0	0.0	0.0
$Adam\ \beta_1$	0.999	0.999	0.999	0.999
EMA decay rate	0.9	0.9	0.95	0.95
Time Distribution	$p_{EDM}(t)$(B.6)	$p_{EDM}(t)$(B.6)	$p_{EDM}(t)$(B.6)	$p_{EDM}(t)$(B.6)
Weighting	$\lambda_{EDM}(t)$(B.8)	1	$\lambda_{EDM}(t)$(B.8)	1
Number of GPUs	8×H800-80G	8×H800-80G	8×H100-80G	8×H800-80G

With the optimal setting and EDM formulation, we can rewrite our algorithm in an EDM style in Algorithm 3.

Algorithm 3: Diff-Instruct* Pseudo Code under EDM formulation.

Input: prompt dataset $\mathcal{C}$, generator $g_\theta(\boldsymbol{x}_0|\boldsymbol{z},\boldsymbol{c})$, prior distribution p_z, reward model $r(\boldsymbol{x},\boldsymbol{c})$, reward scale α_{rew}, CFG scale α_{cfg}, reference EDM denoiser model $\boldsymbol{d}_{ref}(\boldsymbol{x}_t|c,\boldsymbol{c})$, assistant EDM denoiser $\boldsymbol{d}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})$, forward diffusion $p(\boldsymbol{x}_t|\boldsymbol{x}_0)$ (2.1), assistant EDM denoiser updates rounds K_{TA}, time distribution $\pi(t)$, diffusion model weighting $\lambda(t)$, generator IKL loss weighting $w(t)$.

while *not converge* **do**

fix θ, update ψ for K_{TA} rounds by

1. sample prompt $\boldsymbol{c}\sim\mathcal{C}$; sample time $t\sim\pi(t)$; sample $\boldsymbol{z}\sim p_z(\boldsymbol{z})$;
2. generate fake data: $\boldsymbol{x}_0=\mathrm{sg}[g_\theta(\boldsymbol{z},\boldsymbol{c})]$; sample noisy data: $\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$;
3. update ψ by minimizing loss: $\mathcal{L}(\psi)=\lambda(t)\|\boldsymbol{d}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})-\boldsymbol{x}_0\|_2^2$;

fix ψ, update θ using StaD:

1. sample prompt $\boldsymbol{c}\sim\mathcal{C}$; sample time $t\sim\pi(t)$; sample $\boldsymbol{z}\sim p_z(\boldsymbol{z})$;
2. generate fake data: $\boldsymbol{x}_0=g_\theta(\boldsymbol{z},\boldsymbol{c})$; sample noisy data: $\boldsymbol{x}_t\sim p_t(\boldsymbol{x}_t|\boldsymbol{x}_0)$;
3. explicit reward: $\mathcal{L}_{rew}(\theta)=-\alpha_{rew}r(\boldsymbol{x}_0,\boldsymbol{c})$;
4. CFG reward: $\mathcal{L}_{cfg}(\theta)=\alpha_{cfg}\cdot w(t)\big\{\boldsymbol{d}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t,\boldsymbol{c})-\boldsymbol{d}_{ref}(\mathrm{sg}[\boldsymbol{x}_t]|t,\varnothing)\big\}^T\boldsymbol{x}_t$;
5. score-regularization: $\mathcal{L}_{reg}(\theta)=-w(t)\big\{\mathbf{d}'(\boldsymbol{d}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})-\boldsymbol{d}_{ref}(\boldsymbol{x}_t|t,\boldsymbol{c}))\big\}^T\big\{\boldsymbol{d}_\psi(\boldsymbol{x}_t|t,\boldsymbol{c})-\boldsymbol{x}_0\big\}$;
6. update θ by minimizing DI* loss: $\mathcal{L}_{DI*}(\theta)=\mathcal{L}_{rew}(\theta)+\mathcal{L}_{cfg}(\theta)+\mathcal{L}_{reg}(\theta)$;

end

return θ,ψ.

B.6 PYTORCH STYLE PSEUDO-CODE OF SCORE IMPLICIT MATCHING

In this section, we give a PyTorch style pseudo-code for algorithm 3.

```
import torch
import torch.nn as nn
import torch.optim as optim
import copy

use_cfg = True
use_reward = True

# Initialize generator G
G = Generator()

## load teacher DM
Drf = DiffusionModel().load('/path_to_ckpt').eval().requires_grad_(False)
Dta = copy.deepcopy(Drf) ## initialize online DM with teacher DM
r = RewardModel() if use_reward else None

# Define optimizers
opt_G = optim.Adam(G.parameters(), lr=0.001, betas=(0.0, 0.999))
opt_Sta = optim.Adam(Dta.parameters(), lr=0.001, betas=(0.0, 0.999))

# Training loop
while True:
    ## update Dta
    Dta.train().requires_grad_(True)
    G.eval().requires_grad_(False)

    ## update assistant diffusion
    prompt = batch['prompt']
    z = torch.randn((1024, 4, 64, 64), device=G.device)
    with torch.no_grad():
```

```
        fake_x0 = G(z,prompt)

    sigma = torch.exp(2.0*torch.randn([1,1,1,1], device=fake_x0.device) - 2.0)
    noise = torch.randn_like(fake_x0)
    fake_xt = fake_x0 + sigma*noise
    pred_x0 = Dta(fake_xt, sigma, prompt)

    weight = compute_diffusion_weight(sigma)

    batch_loss = weight * (pred_x0 - fake_x0)**2
    batch_loss = batch_loss.sum([1,2,3]).mean()

    optimizer_Dta.zero_grad()
    batch_loss.backward()
    optimizer_Dta.step()


    ## update G
    Dta.eval().requires_grad_(False)
    G.train().requires_grad_(True)

    prompt = batch['prompt']
    z = torch.randn((1024, 4, 64, 64), device=G.device)
    fake_x0 = G(z, prompt)

    sigma = torch.exp(2.0*torch.randn([1,1,1,1], device=fake_x0.device) - 2.0)
    noise = torch.randn_like(fake_x0)
    fake_xt = fake_x0 + sigma*noise

    with torch.no_grad():
        if use_cfg:
            cfg_vector = (Drf(fake_xt, sigma, prompt) - Drf(fake_xt, sigma, None)
        else:
            cfg_vector = None

        pred_x0_rf = Drf(fake_xt, sigma, prompt)
        pred_x0_ta = Dta(fake_xt, sigma, prompt)

    denoise_diff = pred_x0_ta - pred_x0_rf
    adp_wgt = torch.sqrt(denoise_diff.square().sum([1,2,3], keepdims=True) + phuber_c**2)
    weight = compute_G_weight(sigma, denoise_diff)

    # compute score regularization loss
    batch_loss = weight * denoise_diff * (fake_D_yn - D_yn)/adp_wgt

    # compute explicit reward loss if needed
    if use_reward:
        reward_loss = -reward_scale * r(fake_x0, prompt)
        batch_loss += reward_loss

    # compute cfg reward loss if needed
    if use_cfg:
        cfg_reward_loss = cfg_scale * cfg_vector*fake_x0
        batch_loss += cfg_reward_loss

    batch_loss = batch_loss.sum([1,2,3]).mean()

    optimizer_G.zero_grad()
    batch_loss.backward()
```

```
    optimizer_G.step()
```

Listing 2: Pytorch Style Pseudo-code of Diff-Instruct*