One-step Diffusion with Distribution Matching Distillation

Tianwei Yin¹  Michaël Gharbi²  Richard Zhang²  Eli Shechtman²  Frédo Durand¹  William T. Freeman¹  Taesung Park²

¹Massachusetts Institute of Technology  ²Adobe Research

https://tianweiy.github.io/dmd/

Figure 1. Which is which? Among these images, some were generated with baseline Stable Diffusion (SD) [63] (2590ms each), the others with our Diffusion Matching Distillation (DMD) (90ms each). Can you tell which is which? Answers in the footnote. (Non-abbreviated prompts in Appendix G.) Our one-step text-to-image generators provide quality rivaling expensive diffusion models.

Abstract

Diffusion models generate high-quality images but require dozens of forward passes. We introduce Distribution Matching Distillation (DMD), a procedure to transform a diffusion model into a one-step image generator with minimal impact on image quality. We enforce the one-step image generator match the diffusion model at distribution level, by minimizing an approximate KL divergence whose gradient can be expressed as the difference between 2 score functions, one of the target distribution and the other of the synthetic distribution being produced by our one-step generator. The score functions are parameterized as two diffusion models trained separately on each distribution. Combined with a simple regression loss matching the large-scale structure of the multi-step diffusion outputs, our method outperforms all published few-step diffusion approaches, reaching 2.62 FID on ImageNet 64×64 and 11.49 FID on zero-shot COCO-30k, comparable to Stable Diffusion but orders of magnitude faster. Utilizing FP16 inference, our model can generate images at 20 FPS on modern hardware.

1. Introduction

Diffusion models [21, 61, 63, 64, 71, 74] have revolutionized image generation, achieving unprecedented levels of realism and diversity with a stable training procedure. In contrast to GANs [15] and VAEs [34], however, their sampling is a slow, iterative process that transforms a Gaussian noise sample into an intricate image by progressive denoising [21, 74]. This typically requires tens to hundreds of costly neural network evaluations, limiting interactivity in using the generation pipeline as a creative tool.

To accelerate sampling speed, previous methods [42, 43, 47, 48, 51, 65, 75, 90, 91] distill the noise–image mapping, discovered by the original multi-step diffusion sampling, into a single-pass student network. However, fitting such a high-dimensional, complex mapping is certainly a demanding task. A challenge is the expensive cost of run-
ning the full denoising trajectory, just to realize one loss computation of the student model. Recent methods mitigate this by progressively increasing the sampling distance of the student, without running the full denoising sequence of the original diffusion [3, 16, 42, 43, 51, 65, 75]. However, the performance of distilled models still lags behind the original multi-step diffusion model.

In contrast, rather than enforcing correspondences between noise and diffusion-generated images, we simply enforce that the student generations look indistinguishable from the original diffusion model. At high level, our goal shares motivation with other distribution-matching generative models, such as GMMN [39] or GANs [15]. Still, despite their impressive success in creating realistic images [27, 30], scaling up the model on the general text-to-image data has been challenging [26, 62, 87]. In this work, we bypass the issue by starting with a diffusion model that is already trained on large-scale text-to-image data. Concretely, we finetune the pretrained diffusion model to learn not only the data distribution, but also the fake distribution that is being produced by our distilled generator. Since diffusion models are known to approximate the score functions on diffused distributions [23, 73], we can interpret the denoised diffusion outputs as gradient directions for making an image “more realistic”, or if the diffusion model is learned on the fake images, “more fake”. Finally, the gradient update rule for the generator is concocted as the difference of the two, nudging the synthetic images toward higher realism and lower fakeness. Previous work [80], in a method called Variational Score Distillation, shows that modeling the real and fake distributions with a pretrained diffusion model is also effective for test-time optimization of 3D objects. Our insight is that a similar approach can instead train an entire generative model.

Furthermore, we find that pre-computing a modest number of the multi-step diffusion sampling outcomes and enforcing a simple regression loss with respect to our one-step generation serves as an effective regularizer in the presence of the distribution matching loss. Moreover, the regression loss ensures our one-step generator aligns with the teacher model (see Figure 6), demonstrating potential for real-time design previews. Our method draws upon inspiration and insights from VSD [80], GANs [15], and pix2pix [24], showing that by (1) modeling real and fake distributions with diffusion models and (2) using a simple regression loss to match the multi-step diffusion outputs, we can train a one-step generative model with high fidelity.

We evaluate models trained with our Distribution Matching Distillation procedure (DMD) across various tasks, including image generation on CIFAR-10 [36] and ImageNet 64×64 [8], and zero-shot text-to-image generation on MS COCO 512×512 [40]. On all benchmarks, our one-step generator significantly outperforms all published few-steps diffusion methods, such as Progressive Distillation [51, 65], Rectified Flow [42, 43], and Consistency Models [48, 75]. On ImageNet, DMD reaches FIDs of 2.62, an improvement of 2.4× over Consistency Model [75]. Employing the identical denoiser architecture as Stable Diffusion [63], DMD achieves a competitive FID of 11.49 on MS-COCO 2014-30k. Our quantitative and qualitative evaluations show that the images generated by our model closely resemble the quality of those generated by the costly Stable Diffusion model. Importantly, our approach maintains this level of image fidelity while achieving a 100× reduction in neural network evaluations. This efficiency allows DMD to generate 512 × 512 images at a rate of 20 FPS when utilizing FP16 inference, opening up a wide range of possibilities for interactive applications.

2. Related Work

Diffusion Model Diffusion models [2, 21, 71, 74] have emerged as a powerful generative modeling framework, achieving unparalleled success in diverse domains such as image generation [61, 63, 64], audio synthesis [6, 35], and video generation [11, 22, 70]. These models operate by progressively transforming noise into coherent structures through a reverse diffusion process [72, 74]. Despite state-of-the-art results, the inherently iterative procedure of diffusion models entails a high and often prohibitive computational cost for real-time applications. Our work builds upon leading diffusion models [31, 63] and introduces a simple distillation pipeline that reduces the multi-step generative process to a single forward pass. Our method is universally applicable to any diffusion model with deterministic sampling [31, 72, 74].

Diffusion Acceleration Accelerating the inference process of diffusion models has been a key focus in the field, leading to the development of two types of approaches. The first type advances fast diffusion samplers [31, 41, 45, 46, 90], which can dramatically reduce the number of sampling steps required by pre-trained diffusion models—from a thousand down to merely 20-50. However, a further reduction in steps often results in a catastrophic decrease in performance. Alternatively, diffusion distillation has emerged as a promising avenue for further boosting speed [3, 16, 42, 47, 51, 65, 75, 82, 91]. They frame diffusion distillation as knowledge distillation [19], where a student model is trained to distill the multi-step outputs of the original diffusion model into a single step. Luhman et al. [47] and DSN [92] proposed a simple approach of pre-computing the denoising trajectories and training the student model with a regression loss in pixel space. However, a significant challenge is the expensive cost of running the full denoising trajectory for each realization of the loss function. To address this issue, Progressive
Distribution Matching Gradient Computation

Figure 2. Method overview. We train one-step generator $G_θ$ to map random noise $z$ into a realistic image. To match the multi-step sampling outputs of the diffusion model, we pre-compute a collection of noise–image pairs, and occasionally load the noise from the collection and enforce LPIPS [88] regression loss between our one-step generator and the diffusion output. Furthermore, we provide distribution matching gradient $\nabla_\theta D_{KL}$ to the fake image to enhance realism. We inject a random amount of noise to the fake image and pass it to two diffusion models, one pretrained on the real data and the other continually trained on the fake images with a diffusion loss, to obtain its denoised versions. The denoising scores (visualized as mean prediction in the plot) indicate directions to make the images more realistic or fake. The difference between the two represents the direction toward more realism and less fakeness and is backpropagated to the one-step generator.

Distillation (PD) [51, 65] train a series of student models that halve the number of sampling steps of the previous model. InstaFlow [42, 43] progressively learn straighter flows on which the one step prediction maintains accuracy over a larger distance. Consistency Distillation (CD) [75], TRACT [3], and BOOAT [16] train a student model to match its own output at a different timestep on the ODE flow, which in turn is enforced to match its own output at yet another timestep. In contrast, our method shows that the simple approach of Luhman et al. and DSNO to pre-compute the diffusion outputs is sufficient, once we introduce distribution matching as the training objective.

Distribution Matching Recently, a few classes of generative models have shown success in scaling up to complex datasets by recovering samples that are corrupted by a predefined mechanism, such as noise injection [21, 61, 64] or token masking [5, 60, 86]. On the other hand, there exist generative methods that do not rely on sample reconstruction as the training objective. Instead, they match the synthetic and target samples at a distribution level, such as GMM [10, 39] or GANs [15]. Among them, GANs have shown unprecedented quality in realism [4, 26–28, 30, 67], particularly when the GAN loss can be combined with task-specific, auxiliary regression losses to mitigate training instability, ranging from paired image translation [24, 54, 79, 89] to unpaired image editing [37, 55, 94]. Still, GANs are a less popular choice for text-guided synthesis, as careful architectural design is needed to ensure training stability at large scale [26].

Lately, several works [1, 12, 85] drew connections between score-based models and distribution matching. In particular, ProlificDreamer [80] introduced Variational Score Distillation (VSD), which leverages a pretrained text-to-image diffusion model as a distribution matching loss. Since VSD can utilize a large pretrained model for unpaired settings [17, 58], it showed impressive results at particle-based optimization for text-conditioned 3D synthesis. Our method refines and extends VSD for training a deep generative neural network for distilling diffusion models. Furthermore, motivated by the success of GANs in image translation, we complement the stability of training with a regression loss. As a result, our method successfully attains high realism on a complex dataset like LAION [69]. Our method is different from recent works that combine GANs with diffusion [68, 81–83], as our formulation is not grounded in GANs. Our method shares motivation with concurrent works [50, 84] that leverage the VSD objective to train a generator, but differs in that we specialize the method for diffusion distillation by introducing regression loss and showing state-of-the-art results for text-to-image tasks.

3. Distribution Matching Distillation

Our goal is to distill a given pretrained diffusion denoiser, the base model, $μ_{base}$, into a fast “one-step” image generator, $G_θ$, that produces high-quality images without the costly iterative sampling procedure (Sec. 3.1). While we wish to produce samples from the same distribution, we do not necessarily seek to reproduce the exact mapping.
3.2. Distribution Matching Loss

Ideally, we would like our fast generator to produce samples that are indistinguishable from real images. Inspired by the ProlificDreamer [80], we minimize the Kullback–Leibler (KL) divergence between the real and fake image distributions, $p_{\text{real}}$ and $p_{\text{fake}}$, respectively:

$$D_{KL}(p_{\text{fake}} \parallel p_{\text{real}}) = \mathbb{E}_{x \sim p_{\text{fake}}} \left( \log \frac{p_{\text{fake}}(x)}{p_{\text{real}}(x)} \right) = \mathbb{E}_{x \sim \mathcal{N}(0,1)}_{x \sim G_{\theta}(z)} \left( -\log p_{\text{real}}(x) - \log p_{\text{fake}}(x) \right).$$  

(1)

Computing the probability densities to estimate this loss is generally intractable, but we only need the gradient with respect to $\theta$ to train our generator by gradient descent. **Gradient update using approximate scores.** Taking the gradient of Eq. (1) with respect to the generator parameters:

$$\nabla_{\theta} D_{KL} = \mathbb{E}_{z \sim \mathcal{N}(0,1)}_{x \sim G_{\theta}(z)} \left[ - (s_{\text{real}}(x) - s_{\text{fake}}(x)) \frac{dG}{d\theta} \right],$$  

(2)

where $s_{\text{real}}(x) = \nabla_x \log p_{\text{real}}(x)$, $s_{\text{fake}}(x) = \nabla_x \log p_{\text{fake}}(x)$ are the scores of the respective distributions. Intuitively, $s_{\text{real}}$ moves $x$ toward the modes of $p_{\text{real}}$, and $-s_{\text{fake}}$ spreads them apart, as shown in Figure 3(a, b). Computing this gradient is still challenging for two reasons: first, the scores diverge for samples with low probability — in particular $p_{\text{real}}$ vanishes for fake samples, and second, our intended tool for estimating score, namely the diffusion models, only provide scores of the diffused distribution. Score-SDE [73, 74] provides an answer to these two issues.

By perturbing the data distribution with random Gaussian noise of varying standard deviations, we create a family of “blurred” distributions that are fully-supported over the ambient space, and therefore overlap, so that the gradient in Eq. (2) is well-defined (Figure 4). Score-SDE then shows that a trained diffusion model approximates the score function of the diffused distribution.

Accordingly, our strategy is to use a pair of diffusion denoisers to model the scores of the real and fake distributions after Gaussian diffusion. With slight abuse of notation, we define these as $s_{\text{real}}(x_t, t)$ and $s_{\text{fake}}(x_t, t)$, respectively. Diffused sample $x_t \sim q(x_t|x)$ is obtained by adding noise to generator output $x = G_{\theta}(z)$ at diffusion time step $t$:

$$q_t(x_t|x) \sim \mathcal{N} \left( \alpha_t x; \sigma_t^2 I \right),$$  

(3)

where $\alpha_t$ and $\sigma_t$ are from the diffusion noise schedule. **Real score.** The real distribution is fixed, corresponding to the training images of the base diffusion model, so we model its score using a fixed copy of the pretrained diffusion model $\mu_{\text{base}}(z, t)$. The score given a diffusion model is given by Song et al. [74]:

$$s_{\text{real}}(x_t, t) = - \frac{x_t - \alpha_t \mu_{\text{base}}(x_t, t)}{\sigma_t^2},$$  

(4)
Dynamically-learned fake score. We derive the fake score function, in the same manner as the real score case:

\[
s_{\text{fake}}(x_t, t) = \frac{x_t - \alpha t \mu_{\text{fake}}}(x_t, t)}{\sigma_t^2},
\]

However, as the distribution of our generated samples changes throughout training, we dynamically adjust the fake diffusion model \( \mu_{\text{fake}} \) to track these changes. We initialize the fake diffusion model from the pretrained diffusion model \( \mu_{\text{base}} \), updating parameters \( \phi \) during training, by minimizing a standard denoising objective \([21, 77] \):

\[
L^{\phi}_{\text{denoise}} = \| \mu_{\text{fake}}(x_t, t) - x_0 \|^2_2,
\]

where \( L^{\phi}_{\text{denoise}} \) is weighted according to the diffusion timestep \( t \), using the same weighting strategy employed during the training of the base diffusion model \([31, 63] \).

Distribution matching gradient update. Our final approximate distribution matching gradient is obtained by replacing the exact score in Eq. (2) with those defined by the two diffusion models on the perturbed samples \( x_t \) and taking the expectation over the diffusion time steps:

\[
\nabla_{\theta} D_{KL} \approx \mathbb{E}_{z_t \sim p(z_t), x_t \sim p(x_t)} \left[ w_t \alpha_t \left( s_{\text{fake}}(x_t, t) - s_{\text{real}}(x_t, t) \right) \frac{dG_{\theta}}{d\theta} \right],
\]

where \( z \sim \mathcal{N}(0; I), x = G_{\theta}(z), t = \mathcal{U}(T_{\text{min}}, T_{\text{max}}), \) and \( x_t \sim q_t(x_t | x) \). We include the derivations in Appendix F. Here, \( w_t \) is a time-dependent scalar weight we add to improve the training dynamics. We design the weighting factor to normalize the gradient’s magnitude across different noise levels. Specifically, we compute the mean absolute error across spatial and channel dimensions between the denoised image and the input, setting

\[
w_t = \sigma_t^2 || \mu_{\text{base}}(x_t, t) - x_t ||_1^2,
\]

where \( S \) is the number of spatial locations and \( C \) is the number of channels. In Sec. 4.2, we show that this weighting outperforms previous designs \([58, 80] \). We set \( T_{\text{min}} = 0.02 T \) and \( T_{\text{max}} = 0.98 T \), following DreamFusion \([58] \).

3.3. Regression loss and final objective

The distribution matching objective introduced in the previous section is well-defined for \( t \gg 0 \), i.e., when the generated samples are corrupted with a large amount of noise. However, for a small amount of noise, \( s_{\text{real}}(x_t, t) \) often becomes unreliable, as \( p_{\text{real}}(x_t, t) \) goes to zero. Furthermore, as the score \( \nabla_{\theta} \log(p) \) is invariant to scaling of probability density function \( p \), the optimization is susceptible to mode collapse/dropping, where the fake distribution assigns higher overall density to a subset of the modes. To avoid this, we use an additional regression loss to ensure all modes are preserved; see Figure 3(b), (c).

This loss measures the pointwise distance between the generator and base diffusion model outputs, given the same input noise. Concretely, we build a paired dataset \( D = \{ z, y \} \) of random Gaussian noise images \( z \) and the corresponding outputs \( y \), obtained by sampling the pretrained diffusion model \( \mu_{\text{base}} \) using a deterministic ODE solver \([31, 41, 72] \). In our CIFAR-10 and ImageNet experiments, we utilize the Heun solver from EDM \([31] \), with 18 steps for CIFAR-10 and 256 steps for ImageNet. For the LAION experiments, we use the PNDM \([41] \) solver with 50 sampling steps. We find that even a small number of noise-image pairs, generated using less than 1% of the training compute, in the case of CIFAR10, for example, acts as an effective regularizer. Our regression loss is given by:

\[
\mathcal{L}_{\text{reg}} = \mathbb{E}_{(z, y) \sim D} \ell(G_{\theta}(z), y).
\]

We use Learned Perceptual Image Patch Similarity (LPIPS) \([88] \) as the distance function \( \ell \), following InStaFlow \([43] \) and Consistency Models \([75] \).

Final objective. Network \( \mu_{\text{fake}}^{\phi} \) is trained with \( L^{\phi}_{\text{denoise}} \), which is used to help calculate \( \nabla_{\theta} D_{KL} \). For training \( G_{\theta} \), the final objective is \( D_{KL} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} \), using \( \lambda_{\text{reg}} = 0.25 \) unless otherwise specified. The gradient \( \nabla_{\theta} D_{KL} \) is computed in Eq. (7), and gradient \( \nabla_{\theta} \mathcal{L}_{\text{reg}} \) is computed from Eq. (9) with automatic differentiation. We apply the two losses to distinct data streams: unpaired fake samples for the distribution matching gradient and paired examples described in Section 3.3 for the regression loss. Algorithm 1 outlines the final training procedure. Additional details are provided in Appendix B.

3.4. Distillation with classifier-free guidance

Classifier-Free Guidance \([20] \) is widely used to improve the image quality of text-to-image diffusion models. Our approach also applies to diffusion models that use classifier-free guidance. We first generate the corresponding noise-output pairs by sampling from the guided model to construct the paired dataset needed for regression loss \( \mathcal{L}_{\text{reg}} \). When computing the distribution matching gradient \( \nabla_{\theta} D_{KL} \), we substitute the real score with that derived from the mean

\[
\sigma_t^2 || \mu_{\text{base}}(x_t, t) - x_t ||_1^2.
\]
Algorithm 1: DMD Training procedure

Input: Pretrained real diffusion model \( \mu_{\text{real}} \), paired dataset \( \mathcal{D} = \{x_{\text{ref}}, y_{\text{ref}}\} \)

Output: Trained generator \( G \).

1. // Initialize generator and fake score estimators from pretrained model
2. \( G \leftarrow\) copyWeights(\( \mu_{\text{real}} \)), \( \mu_{\text{fake}} \leftarrow\) copyWeights(\( \mu_{\text{real}} \))
3. while train do
4.   // Generate images
5.     Sample batch \( z \sim \mathcal{N}(0, I)^B \) and \( (x_{\text{ref}}, y_{\text{ref}}) \sim \mathcal{D} \)
6.     \( x \leftarrow G(z) \)
7.     \( x = \text{concat}(x, x_{\text{ref}}) \) if dataset is LAION else \( x \)
8.   // Update generator
9.     \( \mathcal{L}_{\text{KL}} \leftarrow \text{distributionMatchingLoss}(\mu_{\text{real}}, \mu_{\text{fake}}, x) \) // Eq 7
10. \( \mathcal{L}_{\text{reg}} \leftarrow \text{LPIPS}(x_{\text{ref}}, y_{\text{ref}}) \) // Eq 9
11. \( \mathcal{L}_G \leftarrow \mathcal{L}_{\text{KL}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} \)
12. \( G \leftarrow\) update(\( G, \mathcal{L}_G \))
13. // Update fake score estimation model
14. // Sample time step \( t \sim U(0, 1) \)
15. \( z_{\text{ref}} \leftarrow \text{forwardDiffusion}(\text{stopgrad}(x_{\text{ref}}), t) \)
16. \( \mathcal{L}_{\text{denoise}} \leftarrow \text{denoisingLoss}(\mu_{\text{fake}}(z_{\text{ref}}, t), \text{stopgrad}(x_{\text{ref}})) \) // Eq 6
17. \( \mu_{\text{fake}} \leftarrow\) update(\( \mu_{\text{fake}}, \mathcal{L}_{\text{denoise}} \))
18. end while

prediction of the guided model. Meanwhile, we do not modify the formulation for the fake score. We train our one-step generator with a fixed guidance scale.

4. Experiments

We assess the capabilities of our approach using several benchmarks, including class-conditional generation on CIFAR-10 [36] and ImageNet [8]. We use the Fréchet Inception Distance (FID) [18] to compare image quality and CLIP Score [59] to evaluate text-to-image alignment. First, we perform a direct comparison on ImageNet (Sec. 4.1), where our distribution matching distillation substantially outperforms competing distillation methods with identical base diffusion models. Second, we perform detailed ablation studies verifying the effectiveness of our proposed modules (Sec. 4.2). Third, we train a text-to-image model on the LAION-Aesthetic-6.25+ dataset [69] with a classifier-free guidance scale of 3 (Sec. 4.3). In this phase, we distill Stable Diffusion v1.5, and we show that our distilled model achieves FID comparable to the original model, while offering a 30× speed-up. Finally, we train another text-to-image model on LAION-Aesthetic-6+, utilizing a higher guidance value of 8 (Sec. 4.3). This model is tailored to enhance visual quality rather than optimize the FID metric. Quantitative and qualitative analysis confirm that models trained with our distribution matching distillation procedure can produce high-quality images rivaling Stable Diffusion. We describe additional training and evaluation details in the appendix.

4.1. Class-conditional Image Generation

We train our model on class-conditional ImageNet-64x64 and benchmark its performance with competing methods.

Results are shown in Table 1. Our model surpasses established GANs like BigGAN-deep [4] and recent diffusion distillation methods, including the Consistency Model [75] and TRACT [3]. Our method remarkably bridges the fidelity gap, achieving a near-identical FID score (within 0.3) compared to the original diffusion model, while also attaining a 512-fold increase in speed. On CIFAR-10, our class-conditional model reaches a competitive FID of 2.66. We include the CIFAR-10 results in the appendix.

<table>
<thead>
<tr>
<th>Method</th>
<th># Fwd</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>BigGAN-deep [4]</td>
<td>1</td>
<td>4.06</td>
</tr>
<tr>
<td>ADM [9]</td>
<td>250</td>
<td>2.07</td>
</tr>
<tr>
<td>Progressive Distillation [65]</td>
<td>1</td>
<td>15.39</td>
</tr>
<tr>
<td>DFNO [91]</td>
<td>1</td>
<td>7.83</td>
</tr>
<tr>
<td>BOOT [16]</td>
<td>1</td>
<td>16.30</td>
</tr>
<tr>
<td>TRACT [3]</td>
<td>1</td>
<td>7.43</td>
</tr>
<tr>
<td>Meng et al. [51]</td>
<td>1</td>
<td>7.54</td>
</tr>
<tr>
<td>Diff-Instruc [50]</td>
<td>1</td>
<td>5.57</td>
</tr>
<tr>
<td>Consistency Model [75]</td>
<td>1</td>
<td>6.20</td>
</tr>
<tr>
<td>DMD (Ours)</td>
<td>1</td>
<td>2.62</td>
</tr>
<tr>
<td>EDM [1] (Teacher) [31]</td>
<td>512</td>
<td>2.32</td>
</tr>
</tbody>
</table>

Table 1. Sample quality comparison on ImageNet-64×64. Baseline numbers are derived from Song et al. [75]. The upper section of the table highlights popular diffusion and GAN approaches [4, 9]. The middle section includes a list of competing diffusion distillation methods. The last row shows the performance of our teacher model, EDM [1] [31].

4.2. Ablation Studies

We first compare our method with two baselines: one omitting the distribution matching objective and the other missing the regression loss in our framework. Table 2 (left) summarizes the results. In the absence of distribution matching loss, our baseline model produces images that lack realism and structural integrity, as illustrated in the top section of Figure 5. Likewise, omitting the regression loss leads to training instability and a propensity for mode collapse, resulting in a reduced diversity of the generated images. This issue is illustrated in the bottom section of Figure 5.

Table 2 (right) demonstrates the advantage of our proposed sample weighting strategy (Section 3.2). We compare with \( \sigma_3 / \alpha_3 \) and \( \sigma_7^2 / \alpha_3 \), two popular weighting schemes utilized by DreamFusion [58] and ProlificDreamer [80]. Our weighting strategy achieves a healthy 0.9 FID improvement as it normalizes the gradient magnitudes across noise levels and stabilizes the optimization.
DMD (ours) without distribution matching

(a) Qualitative comparison between our model (left) and the baseline model excluding the distribution matching objective (right). The baseline model generates images with compromised realism and structural integrity. Images are generated from the same random seed.

(b) Qualitative comparison between our model (left) and the baseline model omitting the regression loss (right). The baseline model tends to exhibit mode collapse and a lack of diversity, as evidenced by the predominant appearance of the grey car (highlighted with a red square). Images are generated from the same random seed.

Figure 5. Ablation studies of our training loss, including the distribution matching objective (top) and the regression loss (bottom).

Table 2. Ablation study. (left) We ablate elements of our training loss. We show the FID results on CIFAR-10 and ImageNet-64×64. (right) We compare different sample weighting strategies for the distribution matching loss.

<table>
<thead>
<tr>
<th>Training loss</th>
<th>CIFAR</th>
<th>ImageNet</th>
<th>Sample weighting CIFAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Dist. Matching</td>
<td>3.82</td>
<td>9.21</td>
<td>3.60</td>
</tr>
<tr>
<td>w/o Regress. Loss</td>
<td>5.58</td>
<td>5.61</td>
<td>3.71</td>
</tr>
<tr>
<td>DMD (Ours)</td>
<td>2.66</td>
<td>2.62</td>
<td>Eq. 8 (Ours)</td>
</tr>
</tbody>
</table>

Table 3. Sample quality comparison on zero-shot text-to-image generation on MS COCO-30k. Baseline numbers are derived from GigaGAN [26]. The dashed line indicates that the result is unavailable.†Results are evaluated by us using the released models. LCM-LoRA is trained with a guidance scale of 7.5. We use a guidance scale of 3 for all other methods. Latency is measured with a batch size of 1.

4.3. Text-to-Image Generation

We use zero-shot MS COCO to evaluate our model’s performance for text-to-image generation. We train a text-to-image model by distilling Stable Diffusion v1.5 [63] on the LAION-Aesthetics-6.25+ [69]. We use a guidance scale of 3, which yields the best FID for the base Stable Diffusion model. The training takes around 36 hours on a cluster of 72 A100 GPUs. Table 3 compares our model to state-of-the-art approaches. Our method showcases superior performance over StyleGAN-T [67], surpasses all other diffusion acceleration methods, including advanced diffusion solvers [46, 90], and diffusion distillation techniques such as Latent Consistency Models [48, 49], UFOGen [83], and InstaFlow [43]. We substantially close the gap between distilled and base models, reaching within 2.7 FID from Stable Diffusion v1.5, while running approximately 30× faster. With FP16 inference, our model generates images at 20 frames per second, enabling interactive applications.

High guidance-scale diffusion distillation. For text-to-image generation, diffusion models typically operate with a high guidance scale to enhance image quality [57, 63]. To evaluate our distillation method in this high guidance-scale regime, we trained an additional text-to-image model. This model distills SD v1.5 using a guidance scale of 8+[63]. Table 4 benchmarks our approach against various diffusion acceleration methods [46, 49, 90]. Similar to the low guidance model, our one-step generator significantly outperforms competing methods, even when they utilize a four-step sampling process. Qualitative comparisons with competing approaches and the base diffusion model are shown in Figure 6.

5. Limitations

While our results are promising, a slight quality discrepancy persists between our one-step model and finer discretizations of the diffusion sampling path, such as those with 100 or 1000 neural network evaluations. Additionally, our framework fine-tunes the weights of both the fake score
Figure 6. Starting from a pretrained diffusion model, here Stable Diffusion (right), our distribution matching distillation algorithm yields a model that can generate images with much higher quality (left) than previous few-steps generators (middle), with the same speed or faster.

<table>
<thead>
<tr>
<th>Method</th>
<th>Latency (s)</th>
<th>FID</th>
<th>CLIP-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM++ (4 step) [46]</td>
<td>0.26s</td>
<td>22.44</td>
<td>0.309</td>
</tr>
<tr>
<td>UniPC (4 step) [90]</td>
<td>0.26s</td>
<td>23.30</td>
<td>0.308</td>
</tr>
<tr>
<td>LCM-LoRA (1 step) [49]</td>
<td>0.09s</td>
<td>77.90</td>
<td>0.238</td>
</tr>
<tr>
<td>LCM-LoRA (2 step) [49]</td>
<td>0.12s</td>
<td>24.28</td>
<td>0.294</td>
</tr>
<tr>
<td>LCM-LoRA (4 step) [49]</td>
<td>0.19s</td>
<td>23.62</td>
<td>0.297</td>
</tr>
<tr>
<td>DMD (Ours)</td>
<td>0.09s</td>
<td><strong>14.93</strong></td>
<td><strong>0.320</strong></td>
</tr>
<tr>
<td>SDv1.5 (Teacher) [63]</td>
<td>2.59s</td>
<td>13.45</td>
<td>0.322</td>
</tr>
</tbody>
</table>

Table 4. **FID/CLIP-Score comparison on MS COCO-30K.** †Results are evaluated by us. LCM-LoRA is trained with a guidance scale of 7.5. We use a guidance scale of 8 for all the other methods. Latency is measured with a batch size of 1.

Acknowledgements

This work was started while TY was an intern at Adobe Research. We are grateful for insightful discussions with Yilun Xu, Guangxuan Xiao, and Minguk Kang. This work is supported, in part by NSF grants 2105819, 1955864, and 2019786 (IAIFI), by the Singapore DSTA under DST000ECI20300823 (New Representations for Vision), as well as by funding from GIST and Amazon.
References


Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. TPAMI, 2018.


