Mean-Shift Distillation for Diffusion Mode Seeking
-Supplementary Material-

Vikas Thamizharasan
UMass Amherst
vthamizharas@umass.edu

Nikitas Chatzis
NTUA
nchatzis@tuc.gr

Iliyan Georgiev
Adobe Research
igeorgiev@adobe.com

Matthew Fisher
Adobe Research
matfishe@adobe.com

Evangelos Kalogerakis
TU Crete
UMass Amherst
kalogerakis@tuc.gr

Difan Liu
Adobe Research
diliu@adobe.com

Nanxuan Zhao
Adobe Research
nanxuanz@adobe.com

Michal Lukáč
Adobe Research
mike.k.lukac@gmail.com

A: Discussions

Why mode-seeking? The desirability of mode seeking varies between applications. When trying to directly sample images from the trained model, we wish to sample from the full variety of the distribution instead of getting only the mode— we want sampling to interpolate between mode-seeking and mode-covering. Methods like DDIM aim for this. On the other hand, when we are optimizing an image (or using the image as a proxy to optimize, e.g. NeRF parameters), any gradient-based optimization will converge to a set of sparse points - local extrema - where the gradients are zero (if it converges at all). This is the intended use-case for SDS, VSD, SDI, and our method, and in this case, it is not possible in general to have the optimization process converge to a distribution of points. Given that, the best we can guarantee is that the points the process converges to are aligned with the distribution. Mode-seeking is our proposed way of achieving that.

Compatibility with other guidance schemes. In low-dimensional settings (eg, our toy experiments), our method can recover the modes and reconstruct the data distribution well without any guidance (See Figure 1 and 3). This is aided by the fact that the conditional score estimates parameterized as $\epsilon_\theta(z_t, c)$ (predicted noise from the pre-trained network) is good by itself, without guidance i.e. $\tilde{\epsilon}_\theta(z_t, c)$. Empirically, we observe that without guidance, ancestral sampling techniques like DDIM produce samples that lie on the data manifold, albeit with few outliers.

This is not the case in the high-dimensional setting with experiments on Stable Diffusion. Here $\epsilon_\theta(z_t, c)$ samples are noticeably bad and are predominantly outliers. Currently, the best fix is to augment these noise estimates with guidance to produce $\tilde{\epsilon}_\theta(z_t, c)$, the strategy prevalent in sampling algorithms. We inherit these practices when performing distillation.

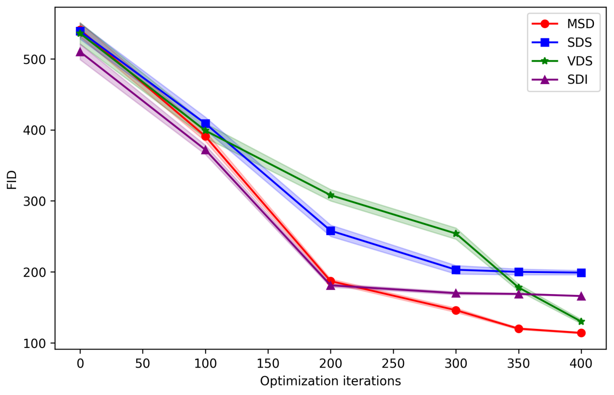

Figure 1. FID vs optimization iterations for text-to-2D generation.

Guidance mechanisms alternative to CFG [1] have been proposed, like Autoguidance [2]. As these methods pair well with DDIM (and other ancestral sampling techniques), we believe the benefits will extend to distillation-based methods like ours. Ultimately from the perspective of distillation, different guidance simply changes the shape of the output distribution but does not fundamentally change the mechanics of diffusion. While we show results using Autoguidance in Table 1, we used CFG in all the remaining experiments as it is more widely used, has hyperparameters (guidance scale) that have been more rigorously tested by the community, and was used in all our baselines (SDS, VSD, and SDI).

B: Implementation Details

Toy distributions in $\mathbb{R}^2$. For the learned denoiser, we use the architecture and training setup used by [2] and similarly represent the densities as mixtures of Gaussians. We use the Adam optimizer [3] and run the optimization procedure for

Input : pre-trained diffusion model $\epsilon_\theta : \mathbb{R}^{d_1\times\cdots\times d_k} \to \mathbb{R}^{d_1\times\cdots\times d_k}$, target parameters $\psi \in \mathbb{R}^d$, condition c, mapping function $g(\psi) : \mathbb{R}^d \to \mathbb{R}^{d_1\times\cdots\times d_k}$, time-dependent functions $w(t), \alpha(t)$, Monte Carlo sample size N.
Output : ψ^*

Algorithm 1: Distillation via SDS

```
for $k = 1, \dots, steps$ do
    $x^k \leftarrow g(\psi)$
    for $i = 1, \dots, N$ do
        $t \leftarrow \mathrm{U}(0, 1)$
        $z_t \leftarrow \alpha(t)x^k + \epsilon_t$
        $y_i \leftarrow w(t)[\epsilon_\theta(z_t, t, c) - \epsilon_t]$
    $\nabla_\psi \mathcal{L}_{SDS} \leftarrow \frac{1}{N}\sum(y_i - x^k)$
    // Backpropagate $\nabla_\psi \mathcal{L}_{SDS}$, update $\psi$
```

Algorithm 2: Distillation via MSD (Ours)

```
function ODESolver($x$, $\lambda$) (eq 12)
    $z_T \leftarrow \mathcal{N}(0, I)$
    for $t = T, \dots, 1$ do
        $z_{t-1} \leftarrow \epsilon_\theta(z_t, t, c) - (x - z_t)/\lambda^2$
    return $z_0$
function ODESolver($x$, $\lambda$, stable) (eq 15)
    $\{z_t^*\}_{t=1}^T \leftarrow inversion(x)$
    $z_T \leftarrow z_T^* + (\epsilon - z_T^*)e^{-\Delta t/\lambda^2}$
    for $t = T, \dots, 1$ do
        $z_{t-1} \leftarrow z_t^* + (\epsilon_\theta(z_t, t, c) - z_t^*)e^{-\Delta t/\lambda^2}$
    return $z_0$
// initialize $\lambda$, set $\lambda_{min}$
for $k = 1, \dots, steps$ do
    $x^k \leftarrow g(\psi)$
    for $i = 1, \dots, N$ do
        $y_i \leftarrow \mathrm{ODESolver}(x^k, \lambda)$
    $\nabla_\psi \mathcal{L}_{MSD} \leftarrow \frac{1}{N}\sum_i^N(y_i - x^k)$
    // Backpropagate $\nabla_\psi \mathcal{L}_{MSD}$, update $\psi$
    // Anneal $\lambda$
    if $\lambda < \lambda_{min}$ then
        // terminate
```

Figure 2. Pseudocode of SDS and our procedure, MSD. We additionally show the numerically stable solver, *ODESolver(. . . , stable)*, which is used for experiments with Stable Diffusion. Note, there is stochasticity in the *ODESolver*.

150 steps with a learning rate of 0.08. For our MSD, we set an initial bandwidth of $0.316 \sim \sqrt{0.1}$ which is linearly decayed over the course of the optimization. For the ideal denoiser, due to it's high cost requirements in time and memory, we instead opt to use a few steps of gradient descent with high learning rate. In addition to optimizing samples, we evaluate both SDS, VSD, SDI, and our MSD gradients across the domain and then numerically integrate them to reconstruct the loss functions they represent.

Runtime and memory usage. For text-to-2D generation, we report FID vs number of optimization iterations in Fig. 1, extending the results in Table 4. Our method converges faster when compared to SDS, VSD, and SDI. The 3D NeRF training stage consumes 27GB GPU memory with 512 rendering resolution and batch size of 1.

Pseudocode. We provide detailed pseudocode, comparing baseline SDS and our MSD in Fig. 2.

C: Integrating Kernel Term

We derive the numerically stable update for the kernel term in equation 12 of the main paper. Consider only the kernel term in equation 12 centered on y:

$$s(z_t) = \frac{y - z_t}{\lambda^2}$$

Expressing this as a first-order linear ODE, we get

$$\frac{dz(t)}{dt} + \frac{z(t)}{\lambda^2} = \frac{y}{\lambda^2}$$

Multiply both sides by $e^{\frac{t}{\lambda^2}}$

$$\frac{dz(t)}{dt}e^{\frac{t}{\lambda^2}} + \frac{z(t)}{\lambda^2}e^{\frac{t}{\lambda^2}} = \frac{y}{\lambda^2}e^{\frac{t}{\lambda^2}} \quad (1)$$

The left-hand side of eq 1 is the derivative of $e^{\frac{t}{\lambda^2}}z(t)$:

$$\frac{d}{dt}\left[e^{\frac{t}{\lambda^2}}z(t)\right] = \frac{y}{\lambda^2}e^{\frac{t}{\lambda^2}}$$

Now, we can integrate both sides from 0 to t for continuous variable t and solve for $z(t)$:

$$\int_0^t \frac{d}{d\tau}\left[e^{\frac{\tau}{\lambda^2}} z(\tau)\right] = y \int_0^t \frac{e^{\frac{\tau}{\lambda^2}}}{\lambda^2} d\tau$$
$$e^{\frac{t}{\lambda^2}} z(t) - z(0) = y e^{\frac{t}{\lambda^2}} - y$$
$$z(t) = y + (z(0) - y) e^{\frac{-t}{\lambda^2}}$$

Advancing one discrete reverse step $\Delta t < 0$ starting at z_t

$$z_{t+\Delta t} = y + (z_t - y) e^{\frac{\Delta t}{\lambda^2}} \tag{2}$$

This gives us the closed-form update shown in equation 14 of the main paper that moves the current iterate z_t exponentially toward y with time constant λ^2. We do not need a numerical integrator, resolving issues when magnitude of the kernel term is high.

D: Ablations and More Results

We provide additional results in Fig. 3 and Fig. 4.

E: List of Prompts

"A DSLR photo of a hamburger"
"A blue jay standing on a large basket of rainbow macarons"
"A DSLR photo of a squirrel dressed as a samurai weighing a katana"
"A DSLR photo of a knight in silver armor"
"Line drawing of a Lizard dressed up like a victorian woman, lineal color"
"A photo of a car made out of sushi"
"A DSLR photo of a tulip"
"A DSLR photo of a Pumpkin head zombie, skinny, highly detailed, photorealistic"
"A watercolor painting of a sparrow, trending on artstation"
"Michelangelo style statue of man sitting on a chair"

F: Licenses

Here we provide the URL, citations and licenses of the open-sourced assets we use in this work.

1. https://github.com/threestudio-project/threestudio [Apache License 2.0].
2. https://github.com/Stability-AI/stablediffusion [MIT License].
3. https://github.com/NVlabs/edm2 [CC BY-NC-SA 4.0].

References

[1] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 1

[2] Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

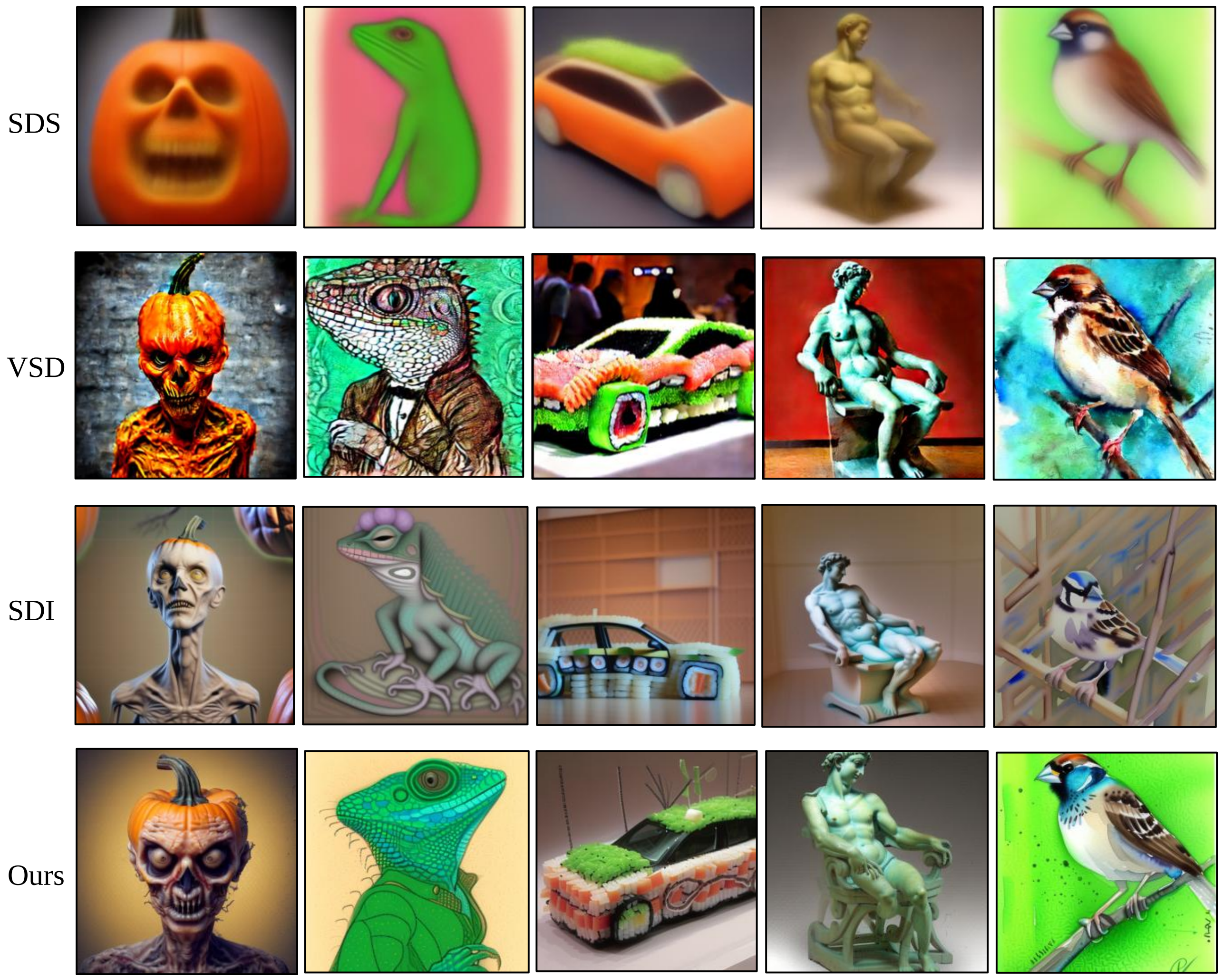

Figure 3. We also show additional results for our method (full), SDI, VSD, and SDS.

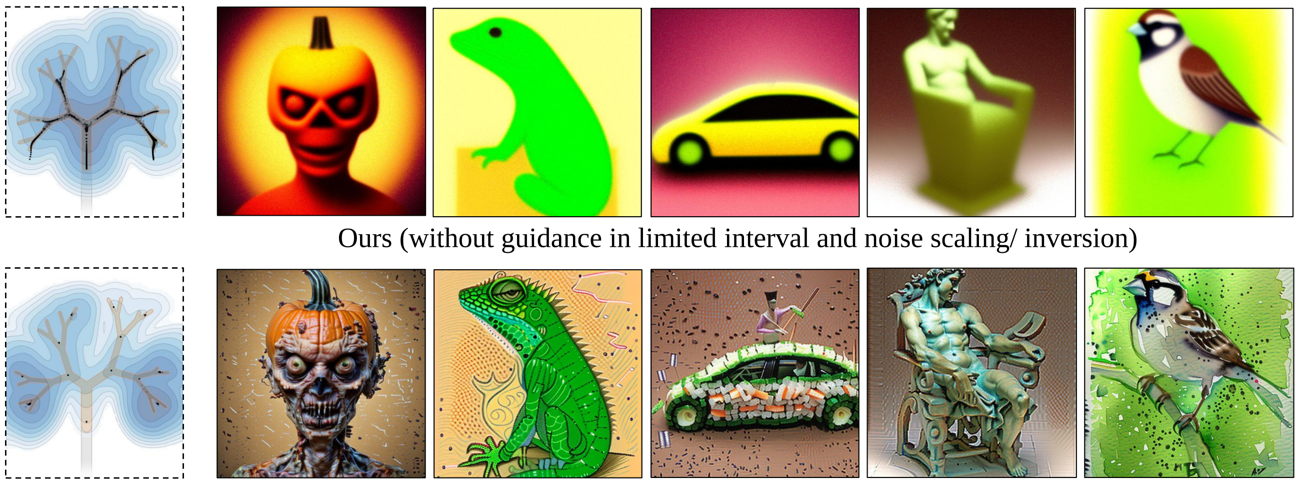
Ours (without guidance in limited interval and noise scaling/ inversion)
Ours (without guidance in limited interval)

Figure 4. We extend Fig. 2 in the main paper with two ablations; applying guidance in the entire denoising trajectory (row 1) and noise scaled sample in the kernel term (row 2).