

DUAL-SOLVER: A GENERALIZED ODE SOLVER FOR DIFFUSION MODELS WITH DUAL PREDICTION

Soochul Park
SteAI
MODULABS
scpark20@gmail.com

Yeon Ju Lee
Korea University
MODULABS
leeyeonju08@korea.ac.kr

ABSTRACT

Diffusion models achieve state-of-the-art image quality. However, sampling is costly at inference time because it requires a large number of function evaluations (NFEs). To reduce NFEs, classical ODE numerical methods have been adopted. Yet, the choice of prediction type and integration domain leads to different sampling behaviors. To address these issues, we introduce Dual-Solver, which generalizes multistep samplers through learnable parameters that continuously (i) interpolate among prediction types, (ii) select the integration domain, and (iii) adjust the residual terms. It retains the standard predictor-corrector structure while preserving second-order local accuracy. These parameters are learned via a classification-based objective using a frozen pretrained classifier (e.g., MobileNet or CLIP). For ImageNet class-conditional generation (DiT, GM-DiT) and text-to-image generation (SANA, PixArt- α), Dual-Solver improves FID and CLIP scores in the low-NFE regime ($3 \leq \text{NFE} \leq 9$) across backbones.

1 INTRODUCTION

Generative modeling aims to learn a data distribution and generate new samples that resemble real data. Classic approaches include autoregressive models that factorize likelihoods over pixels or tokens (Van den Oord et al., 2016; Salimans et al., 2017), variational auto-encoders that optimize an evidence lower bound (Kingma & Welling, 2013; Vahdat & Kautz, 2020), flow-based models that construct exact, invertible density maps (Dinh et al., 2014; Kingma & Dhariwal, 2018), and generative adversarial networks that learn via adversarial training between a generator and a discriminator (Goodfellow et al., 2020; Arjovsky et al., 2017). Diffusion models have emerged as a powerful class of generative models. In seminal work, Sohl-Dickstein et al. (2015) introduced diffusion probabilistic modeling with a forward noising process and a learned reverse process trained by minimizing a KL-divergence objective. Subsequent reformulations (Ho et al., 2020) have streamlined training and inference with simple denoising objectives (e.g., noise or data prediction), leading to state-of-the-art fidelity and robust scaling. Today, diffusion models drive progress across multiple modalities, including images (Dhariwal & Nichol, 2021; Rombach et al., 2022), audio (Kong et al., 2020; Liu et al., 2023), and video (Ho et al., 2022; Kong et al., 2024).

Diffusion models generate samples through iterative denoising steps, making the inference cost scale with the number of function evaluations (NFEs). Leveraging the probability-flow formulation, which casts sampling as an ordinary differential equation (Song et al., 2021b), a number of recent works have proposed ODE-based acceleration methods. Along one axis, classical ODE methods—singlestep Runge–Kutta (Runge, 1895) and multistep Adams–Bashforth (Bashforth & Adams, 1883)—provide off-the-shelf accuracy–NFE trade-offs for a given evaluation budget (Butcher, 2016). Along a second axis, diffusion-dedicated solvers exploit the structure of the denoising dynamics: they approximate noise or data predictions with low-order Taylor expansions or Lagrange interpolation and derive closed-form updates (Lu et al., 2022a;b; Qinsheng & Chen, 2023; Zhao et al., 2023; Xue et al., 2024). Lastly, learned solvers optimize the timestep schedule and other sampling-related parameters (Zhou et al., 2024; Shaul et al., 2023; 2024; Wang et al., 2025). Because these parameters depend on the backbone and dataset, such solvers are typically tailored to specific backbones and settings (e.g., a chosen NFE and guidance scale). Training also

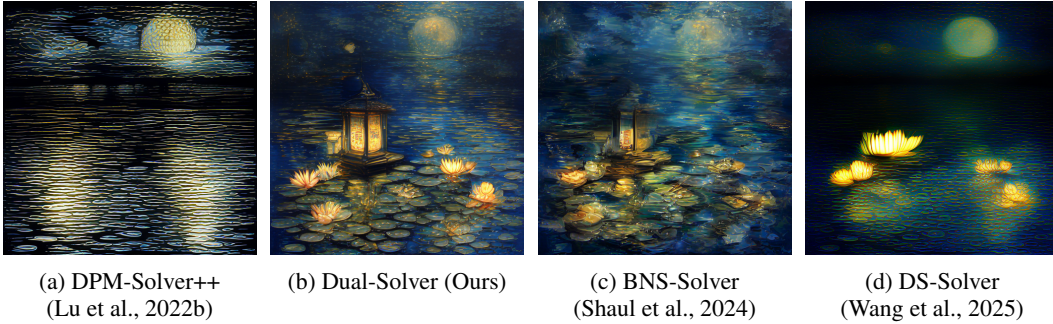


Figure 1: **Sampling results.** SANA (Xie et al., 2024), NFE=3, CFG=4.5. See Fig. 18 for further results.

incurs substantial preparation overhead, as it requires many teacher trajectories or final samples generated using high-NFE sampling. However, compared to the classical methods and dedicated solvers discussed above, learned solvers deliver substantially better sample quality and remain an active area of research. We introduce Dual-Solver, a learned solver for diffusion models with three types of learnable parameters:

- a prediction parameter γ that interpolates among noise, velocity, and data predictions (Sec. 3.1);
- a domain change parameter τ that interpolates between logarithmic and linear domains (Sec. 3.2);
- a residual parameter κ that adjusts the residual term while preserving second-order accuracy (Sec. 3.3).

We further propose a classification-based learning strategy that yields high-fidelity images even in the low-NFE regime (Sec. 5.2). Unlike regression-based learning, which typically requires many target samples generated at high NFEs, our approach requires no target samples. Solver parameters are learned using either pretrained image classifiers (He et al., 2016; Dosovitskiy et al., 2020; Howard et al., 2017) or a zero-shot classifier (Radford et al., 2021). For $3 \leq \text{NFE} \leq 9$, Dual-Solver outperforms prior state-of-the-art solvers (Sec. 6).

2 PRELIMINARIES

2.1 DIFFUSION MODELS

Training process. Diffusion models (Ho et al., 2020; Song et al., 2021b) are trained as follows. A clean sample \mathbf{x}_0 is drawn from the data distribution, and noise is drawn independently from the standard Gaussian distribution. They are then linearly combined with signal rate α_t and noise rate σ_t :

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \quad \text{where } \mathbf{x}_0 \sim p_{\text{data}}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), 0 \leq t \leq T. \quad (1)$$

Here, α_t and σ_t are set by a predefined schedule. Common choices include variance-preserving (VP), with $\alpha_t^2 + \sigma_t^2 = 1$ (Sohl-Dickstein et al., 2015; Ho et al., 2020); variance-exploding (VE), with $\alpha_t = 1$, $\sigma_t \geq 0$ (Song & Ermon, 2019; 2020); and optimal transport (OT), with $\alpha_t = 1 - t$, $\sigma_t = t$, $T = 1$ (Lipman et al., 2022). The backbone is trained to take a noisy sample \mathbf{x}_t and the time t as input and to predict one of the following: the noise $\boldsymbol{\epsilon}$, the clean sample \mathbf{x}_0 (Ho et al., 2020), or the velocity $\mathbf{v}_t = \frac{d\alpha_t}{dt} \mathbf{x}_0 + \frac{d\sigma_t}{dt} \boldsymbol{\epsilon}$ (Lipman et al., 2022). By convention, the backbone parameters are denoted θ , and the predictions are written as $\boldsymbol{\epsilon}_\theta$ (noise), \mathbf{x}_θ (data), and \mathbf{v}_θ (velocity).

Sampling process. The dynamics corresponding to Eq. 1 can be expressed as the following stochastic differential equation (SDE) (Song et al., 2021b):

$$d\mathbf{x}_t = f_t \mathbf{x}_t dt + g_t d\mathbf{w}_t, \quad f_t = \frac{d \log \alpha_t}{dt}, \quad g_t^2 = \frac{d\sigma_t^2}{dt} - 2 \frac{d \log \alpha_t}{dt} \sigma_t^2, \quad (2)$$

where w_t is a standard Wiener process. A probability-flow ODE that shares the same marginal distributions as the SDE at every time t has been proposed (Song et al., 2021b):

$$\frac{d\mathbf{x}_t}{dt} = f_t \mathbf{x}_t - \frac{1}{2} g_t^2 \nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t), \quad \text{where } \nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t) = -\frac{\mathbb{E}[\boldsymbol{\epsilon} | \mathbf{x}_t]}{\sigma_t} \approx -\frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sigma_t}. \quad (3)$$

Given this ODE, one can perform sampling using classical numerical methods beyond Euler, such as singlestep Runge–Kutta (Runge, 1895) and multistep Adams–Bashforth schemes (Bashforth & Adams, 1883). Moreover, several works split the right-hand side into linear and nonlinear parts, and evaluate the nonlinear term using finite-difference approximations (Lu et al., 2022a;b; Zhao et al., 2023) or via Lagrange interpolation (Qinsheng & Chen, 2023; Xue et al., 2024).

2.2 PREDICTION TYPES

Diffusion backbones are typically trained with noise prediction (Ho et al., 2020; Karras et al., 2022; Peebles & Xie, 2023), whereas flow-matching backbones are trained with velocity prediction (Lipman et al., 2022; Chen et al., 2025; Xie et al., 2024). However, noise, velocity, and data predictions can be converted into one another as follows.

$$\mathbf{x}_\theta(\mathbf{x}_t, t) = \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\alpha_t}, \quad \mathbf{v}_\theta(\mathbf{x}_t, t) = \frac{d\alpha_t}{dt} \mathbf{x}_\theta(\mathbf{x}_t, t) + \frac{d\sigma_t}{dt} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \quad (4)$$

Using these relationships, a different prediction type can be used at sampling time from that used during training. For diffusion models, many early samplers use noise prediction (Ho et al., 2020; Song et al., 2021a; Liu et al., 2022; Qinsheng & Chen, 2023; Lu et al., 2022a). More recently, data prediction has also been widely adopted to improve sample quality (Lu et al., 2022b; Xie et al., 2024; Wang et al., 2025). In contrast, flow-matching backbones typically use velocity prediction at sampling time, as in training (Lipman et al., 2022; Chen et al., 2025; Xie et al., 2024; Shaul et al., 2024). Table 1 summarizes the sampling equations in both differential and integral form for each prediction type, derived from Eq. 3 and Eq. 4.

Discretization discrepancy. We ask a simple question: do the prediction types yield the same update? In continuous time, yes; in discrete time, no. Fig. 2 illustrates this discrepancy in two dimensions. For example, applying the Euler method in the λ -domain (half log-SNR) to noise prediction yields $\mathbf{x}_{t_{i+1}}^{\text{noise}} = \frac{\alpha_{t_{i+1}}}{\alpha_{t_i}} [\mathbf{x}_{t_i} - \sigma_{t_i} \Delta\lambda_{t_i} \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, t_i)]$, where $\Delta\lambda_{t_i} := \lambda_{t_{i+1}} - \lambda_{t_i}$. Applying the same method to data prediction and rewriting the result using Eq. 4 gives $\mathbf{x}_{t_{i+1}}^{\text{data}} = \frac{\alpha_{t_{i+1}}}{\alpha_{t_i}} [(1 + \Delta\lambda_{t_i}) e^{-\Delta\lambda_{t_i}} \mathbf{x}_{t_i} - e^{-\Delta\lambda_{t_i}} \sigma_{t_i} \Delta\lambda_{t_i} \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i}, t_i)]$. Since $e^{-\Delta\lambda_{t_i}} = 1 - \Delta\lambda_{t_i} + \frac{1}{2} \Delta\lambda_{t_i}^2 + \dots$, they differ at order $O((\Delta\lambda_{t_i})^2)$. This naturally raises the question of which update is preferable in practice.

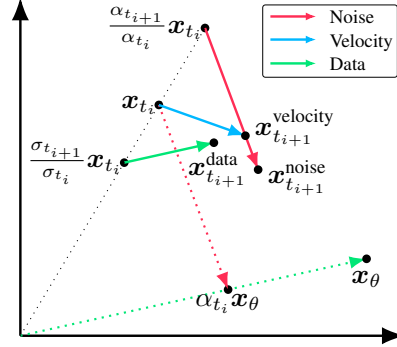


Figure 2: Euler updates for noise, velocity, and data predictions.

Table 1: Differential and integral forms for noise, velocity, and data predictions. (α_t : signal rate, σ_t : noise rate, $\lambda_t := \log(\alpha_t/\sigma_t)$)

	Differential form	Integral form on $[t_i, t_{i+1}]$
Noise	$\frac{d\mathbf{x}_t}{dt} = \frac{d\log \alpha_t}{dt} \mathbf{x}_t - \sigma_t \frac{d\lambda_t}{dt} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$	$\mathbf{x}_{t_{i+1}} = \frac{\alpha_{t_{i+1}}}{\alpha_{t_i}} \mathbf{x}_{t_i} - \alpha_{t_{i+1}} \int_{t_i}^{t_{i+1}} \frac{\sigma_t}{\alpha_t} \frac{d\lambda_t}{dt} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) dt$
Velocity	$\frac{d\mathbf{x}_t}{dt} = \frac{d\alpha_t}{dt} \mathbf{x}_\theta(\mathbf{x}_t, t) + \frac{d\sigma_t}{dt} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$	$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \int_{t_i}^{t_{i+1}} \left[\frac{d\alpha_t}{dt} \mathbf{x}_\theta(\mathbf{x}_t, t) + \frac{d\sigma_t}{dt} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right] dt$
Data	$\frac{d\mathbf{x}_t}{dt} = \frac{d\log \sigma_t}{dt} \mathbf{x}_t + \alpha_t \frac{d\lambda_t}{dt} \mathbf{x}_\theta(\mathbf{x}_t, t)$	$\mathbf{x}_{t_{i+1}} = \frac{\sigma_{t_{i+1}}}{\sigma_{t_i}} \mathbf{x}_{t_i} + \sigma_{t_{i+1}} \int_{t_i}^{t_{i+1}} \frac{\alpha_t}{\sigma_t} \frac{d\lambda_t}{dt} \mathbf{x}_\theta(\mathbf{x}_t, t) dt$

3 DUAL-SOLVER

3.1 DUAL PREDICTION WITH PARAMETER γ

We propose a *dual prediction* scheme that uses both \mathbf{x}_θ and ϵ_θ together. We call it *dual* because it treats \mathbf{x}_θ and ϵ_θ separately, unlike velocity prediction, which bundles them into \mathbf{v}_θ . Moreover, we introduce the following integral formulation parameterized by γ , which interpolates between the integral forms of noise, velocity, and data prediction.

Integral form of dual prediction.

$$\mathbf{x}_{t_{i+1}} = A \mathbf{x}_{t_i} + B \left[\int_{t_i}^{t_{i+1}} C \mathbf{x}_\theta(\mathbf{x}_t, t) dt + \int_{t_i}^{t_{i+1}} D \epsilon_\theta(\mathbf{x}_t, t) dt \right],$$

$$\text{where } \begin{cases} A = (\sigma_{t_{i+1}}/\sigma_{t_i})^\gamma, B = \sigma_{t_{i+1}}^\gamma, C = \frac{d}{dt}(\alpha_t \sigma_t^{-\gamma}), D = \frac{d}{dt}(\sigma_t^{1-\gamma}) \text{ for } \gamma \geq 0, \\ A = (\alpha_{t_{i+1}}/\alpha_{t_i})^{-\gamma}, B = \alpha_{t_{i+1}}^{-\gamma}, C = \frac{d}{dt}(\alpha_t^{1+\gamma}), D = \frac{d}{dt}(\sigma_t \alpha_t^\gamma) \text{ for } \gamma < 0. \end{cases} \quad (5)$$

It has the following properties:

- When $\gamma = -1$, the integral reduces to the noise-prediction form (Table 1).
- When $\gamma = 0$, the integral reduces to the velocity-prediction form (Table 1).
- When $\gamma = 1$, the integral reduces to the data-prediction form (Table 1).

From Eq. 3, we derive the differential form of dual prediction and, by integrating, obtain the integral form given above; the full derivation is provided in Appendix B.1. Another role of γ is to control how much \mathbf{x}_{t_i} contributes to $\mathbf{x}_{t_{i+1}}$: increasing γ decreases the coefficient A multiplying \mathbf{x}_{t_i} , whereas decreasing γ increases A .

3.2 LOG-LINEAR DOMAIN CHANGE WITH PARAMETER τ

Domain change. Let $L : (0, \infty) \rightarrow \mathcal{I}$ be a C^1 diffeomorphism onto an interval $\mathcal{I} \subseteq \mathbb{R}$. To apply a change of variables to the integrals in Eq. 5, we define the following:

$$\begin{cases} u(t) = L(\alpha_t \sigma_t^{-\gamma}), v(t) = L(\sigma_t^{1-\gamma}) \text{ for } \gamma \geq 0, \\ u(t) = L(\alpha_t^{1+\gamma}), v(t) = L(\sigma_t \alpha_t^\gamma) \text{ for } \gamma < 0. \end{cases} \quad (6)$$

By the chain rule, $\frac{d}{dt}L^{-1}(u) = \frac{dL^{-1}(u)}{du} \frac{du}{dt}$ and similarly for v . Thus, we obtain

$$\mathbf{x}_{t_{i+1}} = A \mathbf{x}_{t_i} + B \left[\int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du + \int_{v_i}^{v_{i+1}} \frac{dL^{-1}(v)}{dv} \epsilon_\theta(v) dv \right]. \quad (7)$$

Here, $u_i = u(t_i)$, $u_{i+1} = u(t_{i+1})$ and $v_i = v(t_i)$, $v_{i+1} = v(t_{i+1})$. A and B are as defined in Eq. 5, and $\mathbf{x}_\theta(u) := \mathbf{x}_\theta(\mathbf{x}_{u^{-1}(u)}, u^{-1}(u))$ and $\epsilon_\theta(v) := \epsilon_\theta(\mathbf{x}_{v^{-1}(v)}, v^{-1}(v))$.

Log-linear transform. Previous works (Dockhorn et al., 2022; Qinsheng & Chen, 2023; Zhou et al., 2024) use the linear transform $L(y) = y$ with noise prediction. Because $\frac{d}{du}L^{-1}(u) = 1$, the integrand carries no weighting factor, making it straightforward to develop approximations such as Taylor expansions and Lagrange interpolation. By contrast, other works (Lu et al., 2022a;b; Zhao et al., 2023; Xue et al., 2024) use a logarithmic transform $L(y) = \log y$. Because $\frac{d}{du}L^{-1}(u) = e^u$, the integrand carries an exponential weight. A closed-form approximation can be obtained via an exponential integrator (Hochbruck & Ostermann, 2010) or by using Lagrange interpolation. Motivated by these works, we propose a *log-linear* transform parameterized by a scalar τ .

$$L(y; \tau) = \frac{\log(1 + \tau y)}{\tau}, \quad \tau > 0. \quad (8)$$

This transform is invertible, with inverse $L^{-1}(u; \tau) = (e^{\tau u} - 1)/\tau$; its weighting factor is $\frac{d}{du}L^{-1}(u; \tau) = e^{\tau u}$. Consequently, it has the following properties:

- As $\tau \rightarrow 0^+$: $L(y; \tau) \rightarrow y$, $\frac{d}{du}L^{-1}(u; \tau) \rightarrow 1$.
- When $\tau = 1$: $L(y; \tau) = \log(1 + y)$, $\frac{d}{du}L^{-1}(u; \tau) = e^u$.

We apply the log-linear transform to Eq. 7, allowing separate parameters τ_u and τ_v for the u - and v -integrals, respectively. A comparison of domain-change functions is presented in Sec. 6.2.3.

3.3 APPROXIMATION WITH PARAMETER κ

On the interval $[u_i, u_{i+1}]$, we approximate \mathbf{x}_θ in two ways: (i) the first-order approximation $\mathbf{x}_\theta(u) = \mathbf{x}_\theta(u_i)$; and (ii) the second-order forward-difference approximation $\mathbf{x}_\theta(u) = \mathbf{x}_\theta(u_i) + \frac{\Delta \mathbf{x}_\theta(u_i)}{\Delta u_i}(u - u_i)$. Here, $\mathbf{x}_\theta(u_i) := \mathbf{x}_\theta(\mathbf{x}_{u^{-1}(u_i)}, u^{-1}(u_i))$, $\Delta \mathbf{x}_\theta(u_i) := \mathbf{x}_\theta(u_{i+1}) - \mathbf{x}_\theta(u_i)$, and $\Delta u_i := u_{i+1} - u_i$. We also introduce $K(\Delta u_i; \kappa_u) = \kappa_u (\Delta u_i)^2$, an $\mathcal{O}((\Delta u_i)^2)$ term, to allow additional flexibility in the residual term while preserving local accuracy. κ_u is a real scalar parameter that controls the magnitude of the residual term. Applying the same approximations to ϵ_θ yields the following first-order predictor $\mathbf{x}_{t_{i+1}}^{\text{1st-pred.}}$ and second-order corrector $\mathbf{x}_{t_{i+1}}^{\text{2nd-corr.}}$:

$$\mathbf{x}_{t_{i+1}}^{\text{1st-pred.}} = A \mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i) (\Delta L^{-1}(u_i) + K(\Delta u_i; \kappa_u)) + \epsilon_\theta(v_i) (\Delta L^{-1}(v_i) + K(\Delta v_i; \kappa_v)) \right]. \quad (9)$$

$$\begin{aligned} \mathbf{x}_{t_{i+1}}^{\text{2nd-corr.}} = A \mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{\Delta \mathbf{x}_\theta(u_i)}{2} (\Delta L^{-1}(u_i) + K(\Delta u_i; \kappa_u)) + \right. \\ \left. \epsilon_\theta(v_i) \Delta L^{-1}(v_i) + \frac{\Delta \epsilon_\theta(v_i)}{2} (\Delta L^{-1}(v_i) + K(\Delta v_i; \kappa_v)) \right]. \quad (10) \end{aligned}$$

Here, $\Delta L^{-1}(u_i) := L^{-1}(u_{i+1}) - L^{-1}(u_i)$ and A and B are as defined in Eq. 5. When evaluating $L^{-1}(u_i)$, with α_{t_i} , σ_{t_i} , and γ already known, it can be obtained without writing L^{-1} explicitly. For example, for $\gamma \geq 0$, we have $L^{-1}(u_i) = \alpha_{t_i} \sigma_{t_i}^{-\gamma}$ (see Eq. 6).

Detailed derivations of the predictor and corrector are provided in Appendix B.2, and the corresponding local-accuracy theorem and its proof are given in Appendix C.

4 IMPLEMENTATION DETAILS

Dual-Solver performs sampling using a predictor–corrector scheme (Butcher, 2016) based on the equations developed in the previous section. We detail the sampling scheme in Sec. 4.1 and then present all learnable parameters in Sec. 4.2.

4.1 SAMPLING SCHEME

Alg. 1 describes the sampling procedure of Dual-Solver. Sampling requires a backbone that provides both \mathbf{x}_θ and ϵ_θ ; when only one prediction is available, the other can be obtained via Eq. 4. Given M steps, we use timesteps $\{t_i\}_{i=0}^M$ with $t_0 = T$ and draw the initial noise $\mathbf{x}_{t_0} \sim \mathcal{N}(0, I)$. We also use a list ℓ to store previous evaluations. Empirically, a first-order predictor with a second-order corrector performs best, as shown in Sec. 6.2. At step i , the first-order predictor takes the current state \mathbf{x}_{t_i} and the model evaluations $\{\mathbf{x}_\theta(\mathbf{x}_{t_i}, t_i), \epsilon_\theta(\mathbf{x}_{t_i}, t_i)\}$ to produce a provisional sample $\mathbf{x}'_{t_{i+1}}$. The second-order corrector then combines the evaluations at t_i with fresh evaluations at t_{i+1} , i.e., $\{\mathbf{x}_\theta(\mathbf{x}'_{t_{i+1}}, t_{i+1}), \epsilon_\theta(\mathbf{x}'_{t_{i+1}}, t_{i+1})\}$, to yield the corrected sample $\mathbf{x}_{t_{i+1}}$. At the final step $i = M - 1$, the corrector is not applied.

4.2 LEARNABLE PARAMETERS

For each i -th step, the parameter sets are $\phi_i^{\text{pred}} = \{\gamma_i^{\text{pred}}, \tau_{u,i}^{\text{pred}}, \tau_{v,i}^{\text{pred}}, \kappa_{u,i}^{\text{pred}}, \kappa_{v,i}^{\text{pred}}\}$ and $\phi_i^{\text{corr}} = \{\gamma_i^{\text{corr}}, \tau_{u,i}^{\text{corr}}, \tau_{v,i}^{\text{corr}}, \kappa_{u,i}^{\text{corr}}, \kappa_{v,i}^{\text{corr}}\}$. Thus, each step uses $2 \times 5 = 10$ parameters, except for the last step ($i = M - 1$), which does not use the corrector and therefore has only 5 parameters. Fig. 3 shows the learned parameters for the NFE=5 setting using a DiT (Peebles & Xie, 2023) backbone. Assuming a noise-prediction backbone (the same reasoning applies to data- and velocity-prediction backbones) and a first-order predictor with a second-order corrector (Sec. 4.1), ϕ_i^{pred} and ϕ_i^{corr} determine the coefficients for an update that combines the current state \mathbf{x}_{t_i} and two model evaluations, $\epsilon_\theta(\mathbf{x}'_{t_i}, t_i)$ and $\epsilon_\theta(\mathbf{x}'_{t_{i+1}}, t_{i+1})$, to produce $\mathbf{x}_{t_{i+1}}$. This may seem heavy, but Sec. 6.2.2 shows it is necessary.

In addition to these parameters, we also learn the evaluation times $\{t_i\}_{i=1}^{M-1}$ (with t_0 and t_M fixed), where M denotes the number of steps. Following prior work (Tong et al., 2025; Wang et al., 2025), we employ unnormalized step variables $\{\Delta t'_i\}_{i=0}^{M-1}$ and apply a softmax over $i = 0, \dots, M - 1$ to obtain normalized step sizes $\{\Delta t_i\}_{i=0}^{M-1}$ (nonnegative and summing to one). The timesteps are obtained via a cumulative sum: $t_i = t_0 + (t_M - t_0) \sum_{k=0}^{i-1} \Delta t_k$, $i = 1, \dots, M - 1$.

Algorithm 1 Dual-Solver predictor–corrector sampling (Sec. 4.1)

Require: Diffusion backbone with dual prediction $\{\mathbf{x}_\theta, \epsilon_\theta\}$, timesteps $\{t_i\}_{i=0}^M$, initial noise \mathbf{x}_{t_0} , empty list ℓ , parameters ϕ

- 1: Evaluate $\{\mathbf{x}_\theta(\mathbf{x}_{t_0}, t_0), \epsilon_\theta(\mathbf{x}_{t_0}, t_0)\}$ and add to ℓ
- 2: **for** $i = 0$ **to** $M - 1$ **do**
- 3: $\mathbf{x}'_{t_{i+1}} \leftarrow \text{Predictor}(\mathbf{x}_{t_i}, \ell; \phi_i^{\text{pred}})$
- 4: **if** $i == M - 1$ **then break**
- 5: Evaluate $\{\mathbf{x}_\theta(\mathbf{x}'_{t_{i+1}}, t_{i+1}), \epsilon_\theta(\mathbf{x}'_{t_{i+1}}, t_{i+1})\}$ and add to ℓ
- 6: $\mathbf{x}_{t_{i+1}} \leftarrow \text{Corrector}(\mathbf{x}_{t_i}, \ell; \phi_i^{\text{corr}})$
- 7: **end for**
- 8: **return** \mathbf{x}'_{t_M}

Algorithm 2 Hard-label classification for parameter learning (Sec. 5.2)

Require: Diffusion backbone with parameters θ , VAE decoder \mathcal{D} , pretrained classifier \mathcal{C} , solver \mathcal{S} with parameters ϕ , label dataset \mathcal{Y} , learning rate η

- 1: **while** not converged **do**
- 2: Sample $\mathbf{x}_T \sim \mathcal{N}(0, I)$ \triangleright initial noise
- 3: Sample $y \sim \mathcal{Y}$ \triangleright class label
- 4: $\mathbf{x}_0 \leftarrow \mathcal{S}(\mathbf{x}_T; y, \phi, \theta)$ \triangleright sampling
- 5: $\hat{\mathbf{x}}_0 \leftarrow \mathcal{D}(\mathbf{x}_0)$ \triangleright decoding
- 6: $p \leftarrow \mathcal{C}(\hat{\mathbf{x}}_0)$ \triangleright class probabilities
- 7: $\mathcal{L} \leftarrow \text{CrossEntropy}(p, y)$ \triangleright loss
- 8: $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$ \triangleright parameter update
- 9: **end while**

5 SOLVER PARAMETER LEARNING

In this section, we review existing regression-based parameter learning methods, identify their limitations, and introduce a classification-based approach. Fig. 4 provides a schematic overview of all these methods. We apply them to the proposed Dual-Solver and report FID results for each method in Sec. 6.2.5.

5.1 REGRESSION-BASED PARAMETER LEARNING

In regression-based learning, a solver with trainable parameters is referred to as a student solver, while an existing fixed solver is referred to as a teacher solver. The student is then trained to imitate the behavior of the teacher running at a high NFE. Most prior works adopt *trajectory regression* (Shaul et al., 2023; Zhou et al., 2024; Wang et al., 2025), which compares the trajectories generated by the teacher and the student, or *sample regression* (Shaul et al., 2024), which compares only the final samples. Since comparisons in the trajectory or sample space often show a mismatch with visual perceptual quality, *feature regression* has been proposed (Tong et al., 2025), in which the discrepancy is measured in a feature space using metrics such as LPIPS (Zhang et al., 2018). However, all of these methods require a teacher solver and incur substantial overhead in preparing supervision targets, and they tend to perform poorly in the very low NFE regime (e.g., $\text{NFE} \leq 5$; see Table 5).

5.2 CLASSIFICATION-BASED PARAMETER LEARNING

Beyond feature regression, we consider *soft-label classification*, where we apply a cross-entropy loss between the classifier outputs (probabilities) of the student and the teacher. This approach yields improved results in the low-NFE regime (Table 5), but it still requires a teacher solver to generate the target probabilities.

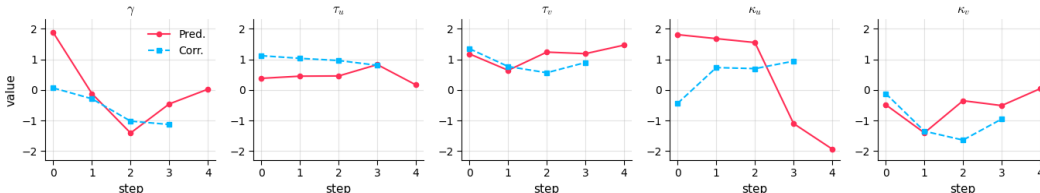


Figure 3: **Learned parameters.** Values of $\{\gamma, \tau_u, \tau_v, \kappa_u, \kappa_v\}$ across sampling steps, learned on DiT (Peebles & Xie, 2023) at NFE = 5. See Figs. 11, 12, 13, and 14 for further results.

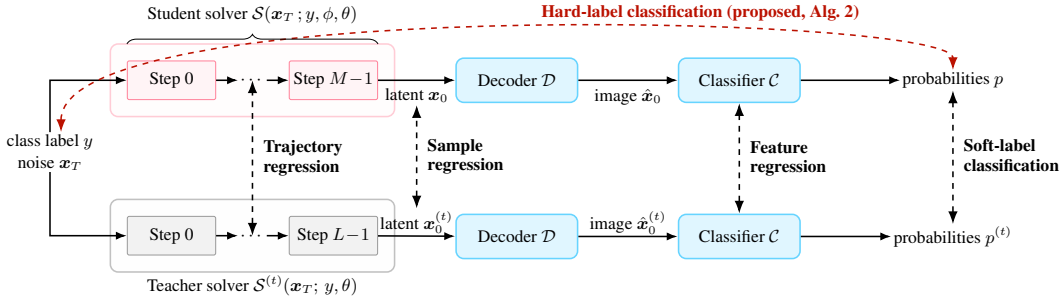


Figure 4: **Solver parameter learning methods.** It schematically illustrates trajectory, sample, and feature regression, as well as soft- and hard-label classification methods.

To remove this dependency, we further propose *hard-label classification*, which compares the student’s classifier outputs (probabilities) against the input class label using a cross-entropy loss, without any teacher solver. The overall training procedure is described in Alg. 2. The sample x_0 generated by the solver \mathcal{S} is passed through the decoder \mathcal{D} and the classifier \mathcal{C} to obtain class probabilities p . We then compute a cross-entropy loss between p and the class label $y \sim \mathcal{Y}$, and update the solver parameters ϕ for all time steps via backpropagation. For text-to-image tasks, the class labels are replaced with text prompts, which can be obtained from datasets such as MSCOCO 2014 (Lin et al., 2014) or MJHQ-30K (Li et al., 2024), and the cross-entropy loss is replaced with CLIP loss (Radford et al., 2021).

Unlike regression to teacher targets, this method focuses on whether the generated samples lie on the correct side of the classifier’s decision boundary, enabling less restrictive training. A potential issue is a mismatch between the distribution learned by the classifier and the ground-truth data distribution, but this can be mitigated by selecting an appropriate classifier. In Sec. 6.2.4, we examine the relationship between the classifier’s accuracy and the resulting FID scores.

6 EXPERIMENTS

We benchmark Dual-Solver against two families of baselines:

- Dedicated solvers: DDIM (Song et al., 2021a), DPM-Solver++ (Lu et al., 2022b).
- Learned solvers: BNS-Solver (Shaul et al., 2024), DS-Solver (Wang et al., 2025).

We select backbones that span diffusion and flow matching, covering both ImageNet (Deng et al., 2009) conditional generation and text-to-image tasks:

- DiT-XL/2-256 \times 256 (Peebles & Xie, 2023): diffusion, ImageNet.
- PixArt- α XL-2-512 (Chen et al., 2023): diffusion, text-to-image.
- GM-DiT 256 \times 256 (Chen et al., 2025): flow matching, ImageNet.
- SANA 600M-512px (Xie et al., 2024): flow matching, text-to-image.

Solver implementations are taken from official sources or reimplemented by us. The backbones can be run via the diffusers library (von Platen et al., 2022). Further details are provided in Appendix D.

6.1 MAIN QUANTITATIVE RESULTS

We evaluate quantitative performance using FID (Heusel et al., 2017) and CLIP score (Radford et al., 2021). For DiT and GM-DiT, FID is computed on 50k images uniformly sampled across the 1,000 ImageNet (Deng et al., 2009) classes. For SANA and PixArt- α , FID and CLIP are computed on the MSCOCO 2014 (Lin et al., 2014) validation set (30k image-caption pairs). The CLIP score is computed as the cosine similarity between text and image features. As described in Sec. 5, we train Dual-Solver with a classification-based objective. For DiT and GM-DiT, MobileNetV3-Large (Howard et al., 2019) is used, and for SANA and PixArt- α , CLIP(RN101) is used. Fig. 5 shows the measured FID and CLIP scores. Across all evaluated NFEs, Dual-Solver outperforms competing solvers on both FID and CLIP for DiT, SANA, and PixArt- α . For GM-DiT, Dual-Solver underperforms at NFE 7–9; however, when trained with a trajectory regression-based objective, it surpasses the baselines at NFE = 8 and 9 (Table 7).

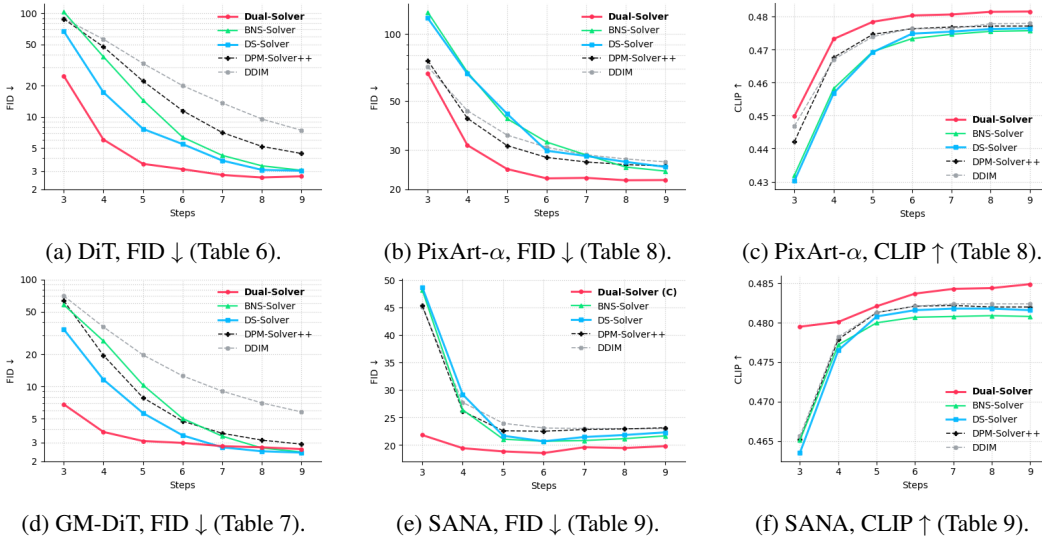


Figure 5: **Main quantitative results.** FID and CLIP score; evaluated on 50k (DiT/GM-DiT) and 30k (SANA/PixArt- α) samples; CFG: DiT=1.5, GM-DiT=1.4, SANA=4.5, PixArt- α =3.5.

6.2 ABLATION STUDY

6.2.1 PREDICTOR-CORRECTOR CONFIGURATIONS

Table 2: Ablation of predictor-corrector configurations on DiT.

NFE	Cross-entropy loss (\downarrow)			
	3	5	7	9
p1	0.667	0.225	0.183	0.175
p1c2	0.574	0.197	0.178	0.173
p2	1.023	0.253	0.222	0.181
p2c2	5.009	0.317	0.203	0.191

We ablate the predictor-corrector configurations: p1 (first-order predictor only), p1c2 (first-order predictor + second-order corrector), p2 (second-order predictor only), and p2c2 (second-order predictor + second-order corrector). The equations for the first-order predictor and second-order corrector are given by Eqs. 9 and 10, respectively, and the equation for the second-order predictor is provided in Eq. 12. As shown in Table 2, p1c2 yields the best performance across NFE = 3, 5, 7, and 9.

6.2.2 PARAMETER SETTINGS FOR (γ, τ, κ)

We ablate the parameterization by fixing selected values or sharing parameters to reduce the degrees of freedom. Setting γ to 1, 0, or -1 recovers data, velocity, and noise prediction, respectively; setting $\tau = 1$ yields the log1p transform; and setting $\kappa = 0$ removes the second-order residual term. We also tie $\tau_u = \tau_v$ and $\kappa_u = \kappa_v$ to share parameters across the integrations for \mathbf{x}_θ and ϵ_θ . As shown in Table 3, leaving all parameters free yields the best performance at NFE = 3 and 5. Configurations that fix or share certain parameters occasionally perform slightly better at NFE = 7 and 9.

Table 3: Ablation of the (γ, τ, κ) parameter settings on DiT.

NFE	Cross-entropy loss (\downarrow)			
	3	5	7	9
$\gamma = 1$ fixed	0.816	0.223	0.182	0.176
$\gamma = 0$ fixed	0.600	0.202	0.183	0.180
$\gamma = -1$ fixed	7.871	7.676	0.238	0.196
$\tau = 1$ fixed	0.601	0.217	0.175	0.178
$\kappa = 0$ fixed	0.944	0.256	0.202	0.190
τ, κ shared	0.667	0.221	0.177	0.169
all learnable	0.574	0.197	0.178	0.173

6.2.3 DOMAIN-CHANGE FUNCTION CHOICE

Table 4: Ablation of Domain Change Functions on DiT.

NFE	FID (\downarrow)			
	3	5	7	9
Type 1	23.38	4.09	3.56	3.64
Type 2	24.91	3.52	2.75	2.67

As candidates for the domain-change function, we consider Type 1, which linearly interpolates between the linear and logarithmic transforms: $L(y; \tau) = (1 - \tau)y + \tau \log y$, and Type 2, defined in Eq. 8. In Table 4, Type 2 achieves noticeably better results at NFE = 5, 7, and 9. According to Eq. 6, the argument y of the function L can become very close to zero; in this regime, the log function diverges, whereas log1p is numerically stable near zero, and we attribute the improved performance to this stability.

6.2.4 CLASSIFICATION MODEL SELECTION

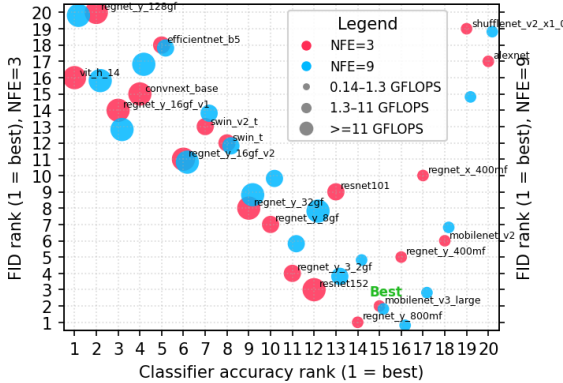


Figure 6: Classifier Accuracy vs. FID on GM-DiT.

A natural question for classification-based learning is which classifier to use. To answer this, we evaluate 20 pretrained classifiers from TorchVision (maintainers & contributors, 2016). Fig. 6 shows FID (on 10k samples) versus accuracy for each model, ordered by rank. The curve exhibits a clear V-shape: as accuracy decreases, FID improves, but beyond ranks 14–16 the FID degrades sharply. This suggests that neither very high nor very low classifier accuracy is optimal; a moderate level is most beneficial for FID. A detailed analysis linking this pattern to precision and recall (Kynkäänniemi et al., 2019) is provided in Appendix E.

6.2.5 PARAMETER LEARNING METHODS

Table 5 reports the results of applying the regression- and classification-based parameter learning methods discussed in Sec. 5 to Dual-Solver. Feature regression is implemented using LPIPS (Zhang et al., 2018), and it generally outperforms trajectory- and sample-space regression. The improvement is particularly pronounced at NFE = 3 and 5. Depending on which classifier (AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2014), or SqueezeNet (Iandola et al., 2016)) is used to extract features, the results vary significantly, underscoring the importance of classifier choice in feature space. Classification-based methods further improve performance over feature regression at NFE = 3 and 5. In particular, the method that uses hard labels achieves the best results across all NFEs. We attribute this to the choice of an appropriate classifier, as discussed in Sec. 6.2.4.

Table 5: Comparison of parameter learning methods on DiT.

NFE	FID (↓)			
	3	5	7	9
Regression-based learning (Sec. 5.1)				
Sample	107.13	11.71	4.60	2.99
Trajectory	100.89	11.59	3.66	2.84
Feature (AlexNet)	47.75	7.24	3.42	2.91
Feature (VGG)	41.58	5.48	3.23	2.88
Feature (SqueezeNet)	44.00	7.07	3.31	2.79
Classification-based learning (Sec. 5.2)				
Soft-label	25.13	4.90	3.37	3.01
Hard-label	24.91	3.52	2.75	2.67

6.3 PARAMETER INTERPOLATION ACROSS NFEs

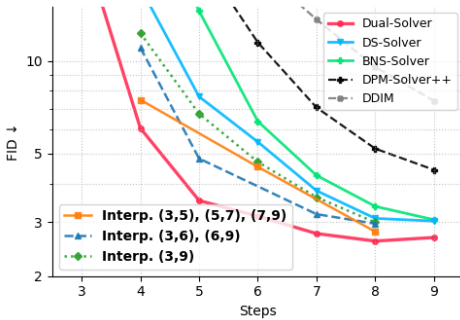


Figure 7: FID results for parameter interpolation across NFEs with DiT.

From the learned parameters of Dual-Solver (Appendix F), we observe that their overall shapes remain similar across different NFEs. Motivated by this, we test the robustness of the learned parameters by applying them to different NFEs. To obtain the parameters for an unseen NFE, we interpolate between the parameters of its two neighboring NFEs. The detailed formulas are provided in Appendix G. Fig. 7 shows the results. *Interp. (3,5), (5,7), (7,9)* denotes that we interpolate between the parameters learned at NFEs (3,5), (5,7), and (7,9) to obtain the parameters for the intermediate NFEs 4, 6, and 8, respectively. The other interpolation schemes are defined analogously. Although these interpolated parameters do not match the performance of parameters directly optimized for each NFE, the gaps are modest, and the resulting FID scores still outperform those of other solvers.

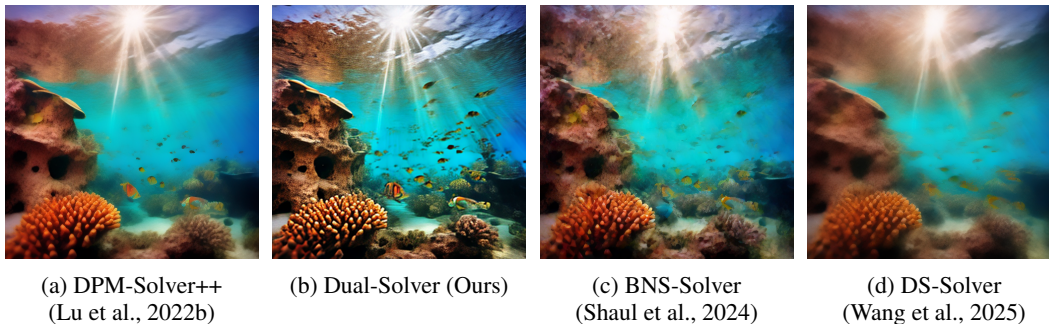


Figure 8: **Sampling results.** PixArt- α (Chen et al., 2023), NFE=5, CFG=3.5. See Fig. 19 for further results.

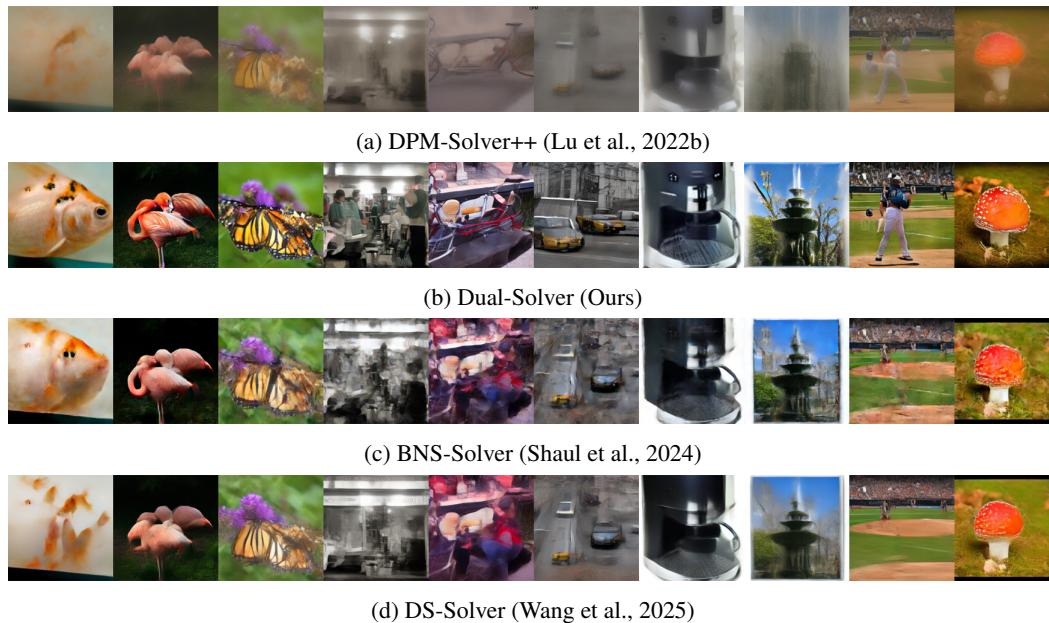


Figure 9: **Sampling results.** GM-DiT (Chen et al., 2025), NFE=3, CFG=1.4. See Fig. 17 for further results.

7 CONCLUSIONS AND LIMITATIONS

This paper introduces Dual-Solver, a predictor–corrector sampler that achieves second-order numerical accuracy. It features per-step learnable parameters (γ, τ, κ) that govern the prediction types, the integration domain, and the second-order residual adjustment. All solver parameters are optimized end-to-end with a classification-based objective using a pretrained image classifier. Across diverse diffusion and flow matching backbones, experiments show substantial improvements over competing solvers in the low-NFE regime ($3 \leq \text{NFE} \leq 9$), as measured by FID and CLIP score. Limitations include the absence of unconditional backbones and a lack of analysis beyond second-order accuracy; both are left for future work.

ACKNOWLEDGEMENTS

We would like to thank Juhee Lee, Jiyub Shin, and Kyungdo Min for their helpful discussions and careful review of the manuscript. This research was partially supported by MODULABS. This work was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (Grant No. RS-2025-25424642).

REFERENCES

- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.
- Francis Bashforth and John Couch Adams. *An Attempt to Test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid: With an Explanation of the Method of Integration Employed in Constructing the Tables*. Cambridge University Press, Cambridge, 1883.
- John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.
- Hansheng Chen, Kai Zhang, Hao Tan, Zexiang Xu, Fujun Luan, Leonidas Guibas, Gordon Wetstein, and Sai Bi. Gaussian mixture flow matching models. *arXiv preprint arXiv:2504.05304*, 2025.
- Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart- α : Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.
- Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.
- Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *Advances in Neural Information Processing Systems*, 35:30150–30166, 2022.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 6626–6637, 2017.
- Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 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.
- Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

- Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.
- Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL <https://doi.org/10.5281/zenodo.5143773>.
- Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.
- Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.
- Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.
- Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation. *arXiv preprint arXiv:2402.17245*, 2024.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.
- Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.
- Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.
- Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022a.
- Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.
- TorchVision maintainers and contributors. Torchvision: Pytorch’s computer vision library. <https://github.com/pytorch/vision>, 2016.
- William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.
- Zhang Qinsheng and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *ICLR*, 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.
- 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.
- Carl Runge. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46:167–178, 1895. URL <http://eudml.org/doc/157756>.
- Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.
- Neta Shaul, Juan Perez, Ricky TQ Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. *arXiv preprint arXiv:2310.19075*, 2023.
- Neta Shaul, Uriel Singer, Ricky TQ Chen, Matthew Le, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke non-stationary solvers for fast sampling of diffusion and flow models. *arXiv preprint arXiv:2403.01329*, 2024.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- 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.
- Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.
- Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in Neural Information Processing Systems*, 33, 2020.
- 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*, 2021b.

- Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion odes. In *International Conference on Learning Representations (ICLR)*, 2025.
- Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.
- Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.
- Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. <https://github.com/huggingface/diffusers>, 2022.
- Shuai Wang, Zexian Li, Tianhui Song, Xubin Li, Tiezheng Ge, Bo Zheng, Limin Wang, et al. Differentiable solver search for fast diffusion sampling. *arXiv preprint arXiv:2505.21114*, 2025.
- Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.
- 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. *Advances in Neural Information Processing Systems*, 36, 2024.
- Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.
- We Zhao, Li Bai, Yong Rao, Jian Zhou, and Jun Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023.
- Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7777–7786, 2024.

CONTENTS

1	Introduction	1
2	Preliminaries	2
2.1	Diffusion Models	2
2.2	Prediction Types	3
3	Dual-Solver	4
3.1	Dual Prediction with Parameter γ	4
3.2	Log-Linear Domain Change with Parameter τ	4
3.3	Approximation with Parameter κ	5
4	Implementation Details	5
4.1	Sampling Scheme	5
4.2	Learnable Parameters	5
5	Solver Parameter Learning	6
5.1	Regression-based Parameter Learning	6
5.2	Classification-based Parameter Learning	6
6	Experiments	7
6.1	Main Quantitative Results	7
6.2	Ablation Study	8
6.3	Parameter Interpolation across NFEs	9
7	Conclusions and Limitations	10
A	LLM Usage	17
B	Derivations	17
B.1	Derivation of Dual Prediction	17
B.2	Derivation of Sampling Equations	18
C	Local Truncation Error	19
C.1	Local Truncation Error of First-Order Predictor	19
C.2	Local Truncation Error of Second-Order Predictor	20
C.3	Local Truncation Error of Second-Order Corrector	21
D	Experimental Details	23
D.1	Setup	23
D.2	Quantitative Results	23

E	Classifier Accuracy vs. Sample Quality	25
F	Learned Parameters	27
G	Details of parameter interpolation across NFEs	31

A LLM USAGE

We disclose that Large Language Models (LLMs) were used as assistive tools for:

- verifying the consistency of mathematical derivations,
- assisting literature search and surfacing related work,
- helping with formal writing (style, grammar, typos),
- drafting figure and table scripts.

All scientific ideas, methods, and results originate from the authors, who take full responsibility for the content of this paper.

B DERIVATIONS

B.1 DERIVATION OF DUAL PREDICTION

Differential Form of Dual Prediction. The differential form corresponding to the integral form in Eq. 5 is given by

$$\frac{d\mathbf{x}_t}{dt} = \beta \mathbf{x}_t + \alpha_t \left(\frac{d \log \alpha_t}{dt} - \beta \right) \mathbf{x}_\theta(\mathbf{x}_t, t) + \sigma_t \left(\frac{d \log \sigma_t}{dt} - \beta \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \quad (11)$$

To derive this form, we start from the probability-flow ODE in Eq. 3:

$$\begin{aligned} \frac{d\mathbf{x}_t}{dt} &= f_t \mathbf{x}_t - \frac{1}{2} g_t^2 \nabla_{\mathbf{x}} \log q(\mathbf{x}_t), \\ \text{where } f_t &= \frac{d \log \alpha_t}{dt}, \quad g_t^2 = \frac{d}{dt} \sigma_t^2 - 2 f_t \sigma_t^2, \quad \nabla_{\mathbf{x}} \log q(\mathbf{x}_t) = - \frac{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sigma_t}. \end{aligned}$$

Substituting f_t , g_t^2 , and $\nabla_{\mathbf{x}} \log q(\mathbf{x}_t)$ yields:

$$\begin{aligned} \frac{d\mathbf{x}_t}{dt} &= \frac{d \log \alpha_t}{dt} \mathbf{x}_t + \frac{1}{2\sigma_t} \left(\frac{d}{dt} \sigma_t^2 - 2 \frac{d \log \alpha_t}{dt} \sigma_t^2 \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \\ &= \frac{d \log \alpha_t}{dt} \mathbf{x}_t + \sigma_t \left(\frac{d \log \sigma_t}{dt} - \frac{d \log \alpha_t}{dt} \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \end{aligned}$$

Introducing an arbitrary $\beta \in \mathbb{R}$, we can rewrite:

$$\begin{aligned} \frac{d\mathbf{x}_t}{dt} &= \beta \mathbf{x}_t + \left(\frac{d \log \alpha_t}{dt} - \beta \right) \mathbf{x}_t + \sigma_t \left(\frac{d \log \sigma_t}{dt} - \frac{d \log \alpha_t}{dt} \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \\ &= \beta \mathbf{x}_t + \left(\frac{d \log \alpha_t}{dt} - \beta \right) (\alpha_t \mathbf{x}_\theta(\mathbf{x}_t, t) + \sigma_t \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) + \sigma_t \left(\frac{d \log \sigma_t}{dt} - \frac{d \log \alpha_t}{dt} \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \\ &= \beta \mathbf{x}_t + \alpha_t \left(\frac{d \log \alpha_t}{dt} - \beta \right) \mathbf{x}_\theta(\mathbf{x}_t, t) + \sigma_t \left(\frac{d \log \sigma_t}{dt} - \beta \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \end{aligned}$$

Thus, we obtain the differential form of *dual prediction*. It is straightforward to verify that choosing $\beta = \frac{d}{dt} \log \alpha_t$ recovers the noise-prediction form, $\beta = \frac{d}{dt} \log \sigma_t$ recovers the data-prediction form, and $\beta = 0$ recovers the velocity-prediction form in Table 1.

Integral Form of Dual Prediction. We next apply the variation-of-constants method to Eq. 11 to obtain the following integral representation:

$$\begin{aligned} \mathbf{x}_{t_{i+1}} &= \exp\left(\int_{t_i}^{t_{i+1}} \beta_u du\right) \mathbf{x}_{t_i} + \\ &\int_{t_i}^{t_{i+1}} \exp\left(\int_s^{t_{i+1}} \beta_u du\right) \left[\alpha_s \left(\frac{d \log \alpha_s}{ds} - \beta_s \right) \mathbf{x}_\theta + \sigma_s \left(\frac{d \log \sigma_s}{ds} - \beta_s \right) \boldsymbol{\epsilon}_\theta \right] ds. \end{aligned}$$

We reparameterize β in terms of a new variable $\gamma \in \mathbb{R}$ as follows:

$$\beta(\gamma) = \begin{cases} \gamma \frac{d \log \sigma_t}{dt} = \gamma \frac{\dot{\sigma}_t}{\sigma_t}, & \gamma \geq 0, \\ -\gamma \frac{d \log \alpha_t}{dt} = -\gamma \frac{\dot{\alpha}_t}{\alpha_t}, & \gamma < 0. \end{cases}$$

Then we obtain the γ -interpolated integral form of dual prediction (Eq. 5).

B.2 DERIVATION OF SAMPLING EQUATIONS

B.2.1 DERIVATION OF FIRST-ORDER PREDICTOR

First, we take a first-order approximation of $\mathbf{x}_\theta(u)$ and $\epsilon_\theta(v)$ in Eq. 7.

$$\mathbf{x}_{t_{i+1}} \approx A\mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \epsilon_\theta(v_i) \Delta L^{-1}(v_i) \right].$$

Here, we write $\mathbf{x}_\theta(u_i) := \mathbf{x}_\theta(\mathbf{x}_{u^{-1}(u_i)}, u^{-1}(u_i))$, and $\Delta L^{-1}(u_i) := L^{-1}(u_{i+1}) - L^{-1}(u_i)$; the definitions for $\epsilon_\theta(v_i)$ and $\Delta L^{-1}(v_i)$ are analogous.

Next, while preserving first-order accuracy, we incorporate the $K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2)$ and $K(\Delta v_i; \kappa_v) = \mathcal{O}((\Delta v_i)^2)$, which yields the first-order predictor of Dual-Solver (Eq. 9).

B.2.2 DERIVATION OF SECOND-ORDER PREDICTOR

First, we approximate $\mathbf{x}_\theta(u)$ and $\epsilon_\theta(v)$ at u_i and v_i by a second-order backward-difference expansion.

$$\begin{aligned} \mathbf{x}_{t_{i+1}} \approx A\mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{\Delta \mathbf{x}_\theta(u_{i-1})}{\Delta u_{i-1}} \int_{u_i}^{u_{i+1}} (u - u_i) \frac{dL^{-1}(u)}{du} du + \right. \\ \left. \epsilon_\theta(v_i) \Delta L^{-1}(v_i) + \frac{\Delta \epsilon_\theta(v_{i-1})}{\Delta v_{i-1}} \int_{v_i}^{v_{i+1}} (v - v_i) \frac{dL^{-1}(v)}{dv} dv \right]. \end{aligned}$$

Using

$$\begin{aligned} \frac{1}{\Delta u_{i-1}} \int_{u_i}^{u_{i+1}} (u - u_i) \frac{dL^{-1}(u)}{du} du \\ = \frac{1}{2r_i^{(u)}} \left(\frac{dL^{-1}(u)}{du} \Big|_{u_i} \Delta u_i + \mathcal{O}((\Delta u_i)^2) \right) \\ = \frac{1}{2r_i^{(u)}} (\Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^2)) \end{aligned}$$

where $r_i^{(u)} := \frac{\Delta u_{i-1}}{\Delta u_i}$ and $r_i^{(v)} := \frac{\Delta v_{i-1}}{\Delta v_i}$, we obtain

$$\begin{aligned} \mathbf{x}_{t_{i+1}} = A\mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{\Delta \mathbf{x}_\theta(u_{i-1})}{2r_i^{(u)}} (\Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^2)) + \right. \\ \left. \epsilon_\theta(v_i) \Delta L^{-1}(v_i) + \frac{\Delta \epsilon_\theta(v_{i-1})}{2r_i^{(v)}} (\Delta L^{-1}(v_i) + \mathcal{O}((\Delta v_i)^2)) \right]. \end{aligned}$$

Next, while preserving second-order accuracy, we incorporate the $K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2)$ and $K(\Delta v_i; \kappa_v) = \mathcal{O}((\Delta v_i)^2)$ residual terms, which yields the following second-order predictor of Dual-Solver.

$$\begin{aligned} \mathbf{x}_{t_{i+1}}^{\text{2nd-pred.}} = A\mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{\Delta \mathbf{x}_\theta(u_{i-1})}{2r_i^{(u)}} (\Delta L^{-1}(u_i) + K(\Delta u_i; \kappa_u)) \right. \\ \left. + \epsilon_\theta(v_i) \Delta L^{-1}(v_i) + \frac{\Delta \epsilon_\theta(v_{i-1})}{2r_i^{(v)}} (\Delta L^{-1}(v_i) + K(\Delta v_i; \kappa_v)) \right] \end{aligned} \quad (12)$$

B.2.3 DERIVATION OF SECOND-ORDER CORRECTOR

First, we approximate $\mathbf{x}_\theta(u)$ and $\epsilon_\theta(v)$ at u_i and v_i by a second-order forward-difference expansion.

$$\mathbf{x}_{t_{i+1}} \approx A\mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i)\Delta L^{-1}(u_i) + \frac{\Delta\mathbf{x}_\theta(u_i)}{\Delta u_i} \int_{u_i}^{u_{i+1}} (u - u_i) \frac{dL^{-1}(u)}{du} du + \epsilon_\theta(v_i)\Delta L^{-1}(v_i) + \frac{\Delta\epsilon_\theta(v_i)}{\Delta v_i} \int_{v_i}^{v_{i+1}} (v - v_i) \frac{dL^{-1}(v)}{dv} dv \right].$$

Using

$$\begin{aligned} & \frac{1}{\Delta u_i} \int_{u_i}^{u_{i+1}} (u - u_i) \frac{dL^{-1}(u)}{du} du \\ &= \frac{1}{2} \left(\left. \frac{dL^{-1}(u)}{du} \right|_{u_i} \Delta u_i + \mathcal{O}((\Delta u_i)^2) \right) \\ &= \frac{1}{2} (\Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^2)) \end{aligned}$$

we obtain

$$\begin{aligned} \mathbf{x}_{t_{i+1}} = A\mathbf{x}_{t_i} + B \left[\mathbf{x}_\theta(u_i)\Delta L^{-1}(u_i) + \frac{\Delta\mathbf{x}_\theta(u_i)}{2} (\Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^2)) \right. \\ \left. + \epsilon_\theta(v_i)\Delta L^{-1}(v_i) + \frac{\Delta\epsilon_\theta(v_i)}{2} (\Delta L^{-1}(v_i) + \mathcal{O}((\Delta v_i)^2)) \right]. \end{aligned}$$

Next, while preserving second-order accuracy, we incorporate the $K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2)$ and $K(\Delta v_i; \kappa_v) = \mathcal{O}((\Delta v_i)^2)$ residual terms, which yields the second-order corrector of Dual-Solver (Eq. 10).

C LOCAL TRUNCATION ERROR

C.1 LOCAL TRUNCATION ERROR OF FIRST-ORDER PREDICTOR

Theorem C.1. *Assume that $\mathbf{x}_\theta(u)$ and $\epsilon_\theta(v)$ are C^1 on $[u_i, u_{i+1}]$ and $[v_i, v_{i+1}]$, respectively. Let $\mathbf{x}_{t_{i+1}}^{\text{exact}}$ denote the exact update in equation 7, and let $\mathbf{x}_{t_{i+1}}^{\text{1st-pred.}}$ denote the first-order predictor defined in equation 9. Then we have*

$$\|\mathbf{x}_{t_{i+1}}^{\text{exact}} - \mathbf{x}_{t_{i+1}}^{\text{1st-pred.}}\| = \mathcal{O}((\Delta u_i)^2 + (\Delta v_i)^2).$$

Proof. From Eq. 7, the exact update can be written as

$$\mathbf{x}_{t_{i+1}}^{\text{exact}} = A\mathbf{x}_{t_i} + B[I_u + I_v],$$

where $I_u = \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du$ and $I_v = \int_{v_i}^{v_{i+1}} \frac{dL^{-1}(v)}{dv} \epsilon_\theta(v) dv$.

Eq. 9 defines the first-order predictor

$$\mathbf{x}_{t_{i+1}}^{\text{1st-pred.}} = A\mathbf{x}_{t_i} + B[I'_u + I'_v],$$

with $I'_u = \mathbf{x}_\theta(u_i) (\Delta L^{-1}(u_i) + K(\Delta u_i; \kappa_u))$ and $I'_v = \epsilon_\theta(v_i) (\Delta L^{-1}(v_i) + K(\Delta v_i; \kappa_v))$.

The accuracy can be obtained by taking a Taylor expansion of I_u and estimating the order of $I_u - I'_u$.

Expansion of I_u

$$\begin{aligned} I_u &= \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du \\ &= \mathbf{x}_\theta(u_i)\Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^2). \end{aligned}$$

Order estimate of $I_u - I'_u$

$$I_u - I'_u = \mathcal{O}((\Delta u_i)^2) - \mathbf{x}_\theta(u_i) K(\Delta u_i; \kappa_u).$$

Since $\mathbf{x}_\theta(u_i) = \mathcal{O}(1)$ and $K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2)$, it follows that

$$\mathbf{x}_\theta(u_i) K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2).$$

Therefore,

$$I_u - I'_u = \mathcal{O}((\Delta u_i)^2).$$

The bound $I_v - I'_v = \mathcal{O}((\Delta v_i)^2)$ can be derived in the same way.

Conclusion

$$\begin{aligned} \mathbf{x}_{t_{i+1}}^{\text{exact}} - \mathbf{x}_{t_{i+1}}^{\text{1st-pred.}} &= B[(I_u - I'_u) + (I_v - I'_v)] \\ &= B[\mathcal{O}((\Delta u_i)^2) + \mathcal{O}((\Delta v_i)^2)] \\ &= \mathcal{O}((\Delta u_i)^2 + (\Delta v_i)^2). \end{aligned}$$

□

C.2 LOCAL TRUNCATION ERROR OF SECOND-ORDER PREDICTOR

Theorem C.2. Assume that $\mathbf{x}_\theta(u)$ and $\epsilon_\theta(v)$ are C^2 on $[u_{i-1}, u_{i+1}]$ and $[v_{i-1}, v_{i+1}]$, respectively. Let $\mathbf{x}_{t_{i+1}}^{\text{exact}}$ denote the exact update in equation 7, and let $\mathbf{x}_{t_{i+1}}^{\text{2nd-pred.}}$ denote the second-order predictor defined in equation 12. Then we have

$$\|\mathbf{x}_{t_{i+1}}^{\text{exact}} - \mathbf{x}_{t_{i+1}}^{\text{2nd-pred.}}\| = \mathcal{O}((\Delta u_i)^3 + (\Delta v_i)^3).$$

Proof. From Eq. 7, the exact update is given by

$$\mathbf{x}_{t_{i+1}}^{\text{exact}} = A\mathbf{x}_{t_i} + B[I_u + I_v],$$

where $I_u = \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du$ and $I_v = \int_{v_i}^{v_{i+1}} \frac{dL^{-1}(v)}{dv} \epsilon_\theta(v) dv$.

Eq. 12 defines the second-order predictor

$$\mathbf{x}_{t_{i+1}}^{\text{2nd-pred.}} = A\mathbf{x}_{t_i} + B[I'_u + I'_v],$$

with $I'_u = \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{\Delta \mathbf{x}_\theta(u_{i-1})}{2r_i^{(u)}} (\Delta L^{-1}(u_i) + K(\Delta u_i; \kappa_u))$ and $I'_v = \epsilon_\theta(v_i) \Delta L^{-1}(v_i) + \frac{\Delta \epsilon_\theta(v_{i-1})}{2r_i^{(v)}} (\Delta L^{-1}(v_i) + K(\Delta v_i; \kappa_v))$.

The accuracy can be obtained by taking a Taylor expansion of I_u and estimating the order of $I_u - I'_u$.

Expansion of I_u

$$\begin{aligned} I_u &= \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} (\mathbf{x}_\theta(u) - \mathbf{x}_\theta(u_i)) du \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \mathbf{x}'_\theta(u_i) \int_{u_i}^{u_{i+1}} (u - u_i) \frac{dL^{-1}(u)}{du} du + \mathcal{O}((\Delta u_i)^3) \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{1}{2} \mathbf{x}'_\theta(u_i) \Delta u_i \Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^3) \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{1}{2r_i^{(u)}} \Delta \mathbf{x}_\theta(u_{i-1}) \Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^3). \end{aligned}$$

Order estimate of $I_u - I'_u$

$$I_u - I'_u = \mathcal{O}((\Delta u_i)^3) - \frac{1}{2r_i^{(u)}} \Delta \mathbf{x}_\theta(u_{i-1}) K(\Delta u_i; \kappa_u).$$

Since $\Delta \mathbf{x}_\theta(u_{i-1}) = \mathcal{O}(\Delta u_i)$ and $K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2)$, it follows that

$$\Delta \mathbf{x}_\theta(u_{i-1}) K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^3).$$

Therefore,

$$I_u - I'_u = \mathcal{O}((\Delta u_i)^3).$$

The bound $I_v - I'_v = \mathcal{O}((\Delta v_i)^3)$ can be derived in the same way.

Conclusion

$$\begin{aligned} \mathbf{x}_{t_{i+1}}^{\text{exact}} - \mathbf{x}_{t_{i+1}}^{\text{2nd-pred.}} &= B[(I_u - I'_u) + (I_v - I'_v)] \\ &= B[\mathcal{O}((\Delta u_i)^3) + \mathcal{O}((\Delta v_i)^3)] \\ &= \mathcal{O}((\Delta u_i)^3 + (\Delta v_i)^3). \end{aligned}$$

□

C.3 LOCAL TRUNCATION ERROR OF SECOND-ORDER CORRECTOR

Theorem C.3. Assume that $\mathbf{x}_\theta(u)$ and $\epsilon_\theta(v)$ are C^2 on $[u_i, u_{i+1}]$ and $[v_i, v_{i+1}]$, respectively. Let $\mathbf{x}_{t_{i+1}}^{\text{exact}}$ denote the exact update in equation 7, and let $\mathbf{x}_{t_{i+1}}^{\text{2nd-corr.}}$ denote the second-order corrector defined in equation 10. Then we have

$$\|\mathbf{x}_{t_{i+1}}^{\text{exact}} - \mathbf{x}_{t_{i+1}}^{\text{2nd-corr.}}\| = \mathcal{O}((\Delta u_i)^3 + (\Delta v_i)^3).$$

Proof. From Eq. 7, the exact update is expressed as

$$\mathbf{x}_{t_{i+1}}^{\text{exact}} = A\mathbf{x}_{t_i} + B[I_u + I_v],$$

where $I_u = \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du$ and $I_v = \int_{v_i}^{v_{i+1}} \frac{dL^{-1}(v)}{dv} \epsilon_\theta(v) dv$.

Eq. 10 defines the second-order corrector

$$\mathbf{x}_{t_{i+1}}^{\text{2nd-corr.}} = A\mathbf{x}_{t_i} + B[I'_u + I'_v],$$

where $I'_u = \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{\Delta \mathbf{x}_\theta(u_i)}{2} (\Delta L^{-1}(u_i) + K(\Delta u_i; \kappa_u))$ and $I'_v = \epsilon_\theta(v_i) \Delta L^{-1}(v_i) + \frac{\Delta \epsilon_\theta(v_i)}{2} (\Delta L^{-1}(v_i) + K(\Delta v_i; \kappa_v))$.

The accuracy can be obtained by taking a Taylor expansion of I_u and estimating the order of $I_u - I'_u$.

Expansion of I_u

$$\begin{aligned} I_u &= \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} \mathbf{x}_\theta(u) du \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \int_{u_i}^{u_{i+1}} \frac{dL^{-1}(u)}{du} (\mathbf{x}_\theta(u) - \mathbf{x}_\theta(u_i)) du \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \mathbf{x}'_\theta(u_i) \int_{u_i}^{u_{i+1}} (u - u_i) \frac{dL^{-1}(u)}{du} du + \mathcal{O}((\Delta u_i)^3) \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{1}{2} \mathbf{x}'_\theta(u_i) \Delta u_i \Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^3) \\ &= \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \frac{1}{2} \Delta \mathbf{x}_\theta(u_i) \Delta L^{-1}(u_i) + \mathcal{O}((\Delta u_i)^3). \end{aligned}$$

Order estimate of $I_u - I'_u$

$$I_u - I'_u = \mathcal{O}((\Delta u_i)^3) - \frac{1}{2} \Delta \mathbf{x}_\theta(u_i) K(\Delta u_i; \kappa_u).$$

Since $\Delta \mathbf{x}_\theta(u_i) = \mathcal{O}(\Delta u_i)$ and $K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^2)$, it follows that

$$\Delta \mathbf{x}_\theta(u_i) K(\Delta u_i; \kappa_u) = \mathcal{O}((\Delta u_i)^3).$$

Therefore,

$$I_u - I'_u = \mathcal{O}((\Delta u_i)^3).$$

The bound $I_v - I'_v = \mathcal{O}((\Delta v_i)^3)$ can be derived in the same way.

Conclusion

$$\begin{aligned} \mathbf{x}_{t_{i+1}}^{\text{exact}} - \mathbf{x}_{t_{i+1}}^{\text{2nd-corr.}} &= B[(I_u - I'_u) + (I_v - I'_v)] \\ &= B[\mathcal{O}((\Delta u_i)^3) + \mathcal{O}((\Delta v_i)^3)] \\ &= \mathcal{O}((\Delta u_i)^3 + (\Delta v_i)^3). \end{aligned}$$

□

D EXPERIMENTAL DETAILS

D.1 SETUP

Environment details. We run all experiments on a single NVIDIA RTX 6000 pro (Driver 575.57.08) under Ubuntu 24.04 with Python 3.11.13, PyTorch 2.8.0, and CUDA 12.9.

Backbone Details. We evaluate DiT-XL/2 (Peebles & Xie, 2023), GM-DiT (Chen et al., 2025), SANA (Xie et al., 2024), and PixArt- α (Chen et al., 2023). All models are obtained via the diffusers library (von Platen et al., 2022) pipelines and run in evaluation mode with bfloat16. The model identifiers are:

- DiT: facebook/DiT-XL-2-256
- GM-DiT: Lakonik/gmflow_imagenet_k8_ema
- SANA: Efficient-Large-Model/Sana_600M_512px_diffusers
- PixArt- α : PixArt-alpha/PixArt-XL-2-512x512

Solver Details. The solvers used in our experiments are the diffusion-dedicated DDIM (Song et al., 2021a), the second-order multistep DPM-Solver++ (Lu et al., 2022b), and the learned solvers BNS-Solver (Shaul et al., 2024) and DS-Solver (Wang et al., 2025). We use DDIM as the first-order counterpart of DPM-Solver++ (as proposed in Lu et al. (2022a)). Implementations of DPM-Solver++ and DS-Solver are taken from their official GitHub repositories^{1,2}, and BNS-Solver is implemented according to the paper.

Learning Details. We train with AdamW (Loshchilov & Hutter, 2017) using $\beta = (0.9, 0.999)$, $\epsilon = 1e-8$, and weight decay 0.01. The learning rate decays from 2×10^{-3} to 1×10^{-4} over 20k steps via cosine annealing (Loshchilov & Hutter, 2016). The batch size is fixed to 10 for all experiments. For regression-based learning, the teacher trajectory is a 200-step DDIM (Song et al., 2021a) method on 1k samples.

Sampling Details. We evaluate all backbones on an NFE grid of 3, 4, 5, 6, 7, 8, 9. For classifier-free guidance (CFG; Ho & Salimans, 2021), we use backbone-specific fixed scales: DiT 1.5, GM-DiT 1.4, SANA 4.5, and PixArt- α 3.5. We compute FID using the publicly released ImageNet training-set statistics (Dhariwal & Nichol, 2021) for ImageNet, and MSCOCO 2014 validation-set statistics for text-to-image (SANA, PixArt- α). CLIP scores are computed with the official RN101 variant (Radford et al., 2021). We generate 50k images for ImageNet and 30k images for text-to-image evaluation.

D.2 QUANTITATIVE RESULTS

Table 6: DiT (Peebles & Xie, 2023): FID (\downarrow) vs. NFE. ImageNet generation, 50k samples.

Method	3	4	5	6	7	8	9
DDIM (Song et al., 2021a)	89.33	56.33	32.91	20.06	13.64	9.55	7.42
DPM-Solver++(Lu et al., 2022b)	88.46	47.64	22.19	11.49	7.06	5.19	4.43
BNS-Solver(Shaul et al., 2024)	103.26	38.20	14.53	6.37	4.25	3.37	3.05
DS-Solver(Wang et al., 2025)	67.31	17.31	7.66	5.46	3.79	3.08	3.02
Dual-Solver (Ours)	24.91	6.05	3.52	3.13	2.75	2.60	2.67

¹<https://github.com/LuChengTHU/dpm-solver>

²<https://github.com/MCG-NJU/NeuralSolver>

Table 7: **GM-DiT** Chen et al. (2025): FID (\downarrow) vs. NFE. ImageNet generation, 50k samples.

Method	3	4	5	6	7	8	9
DDIM (Song et al., 2021a)	70.15	35.98	19.70	12.55	9.03	7.01	5.78
DPM-Solver++(Lu et al., 2022b)	63.24	19.53	7.85	4.74	3.65	3.15	2.89
BNS-Solver(Shaul et al., 2024)	57.88	26.64	10.31	5.02	3.43	2.66	2.44
DS-Solver(Wang et al., 2025)	34.15	11.60	5.64	3.49	2.70	2.48	2.41
Dual-Solver-R (Ours)	45.53	14.43	7.49	3.75	2.77	2.44	2.32
Dual-Solver-C (Ours)	6.81	3.76	3.09	2.97	2.77	2.70	2.60

R = trajectory regression-based; C = classification-based.

Table 8: **PixArt- α** (Chen et al., 2023): FID (\downarrow) and CLIP score (\uparrow) vs. NFE. Text-to-image on MSCOCO 2014 (Lin et al., 2014) with 30k samples.

Method	3	4	5	6	7	8	9
FID (\downarrow)							
DDIM (Song et al., 2021a)	71.37	45.21	35.12	30.92	28.60	27.39	26.58
DPM-Solver++ (Lu et al., 2022b)	76.01	41.77	31.48	27.83	26.56	25.82	25.48
BNS-Solver (Shaul et al., 2024)	125.65	66.94	41.31	32.55	28.55	25.18	24.15
DS-Solver (Wang et al., 2025)	118.01	66.09	43.62	29.74	28.25	26.57	25.22
Dual-Solver (Ours)	66.61	31.61	24.68	22.39	22.51	21.96	22.01
CLIP (RN101, \uparrow)							
DDIM (Song et al., 2021a)	0.4469	0.4670	0.4739	0.4764	0.4763	0.4778	0.4779
DPM-Solver++ (Lu et al., 2022b)	0.4422	0.4676	0.4746	0.4763	0.4768	0.4771	0.4771
BNS-Solver (Shaul et al., 2024)	0.4320	0.4582	0.4694	0.4733	0.4746	0.4755	0.4757
DS-Solver (Wang et al., 2025)	0.4303	0.4568	0.4692	0.4748	0.4754	0.4762	0.4764
Dual-Solver (Ours)	0.4499	0.4732	0.4784	0.4803	0.4806	0.4814	0.4815

Table 9: **SANA** (Xie et al., 2024): FID (\downarrow) and CLIP score (RN101, \uparrow) vs. NFE. Text-to-image on MSCOCO 2014 (Lin et al., 2014) with 30k samples.

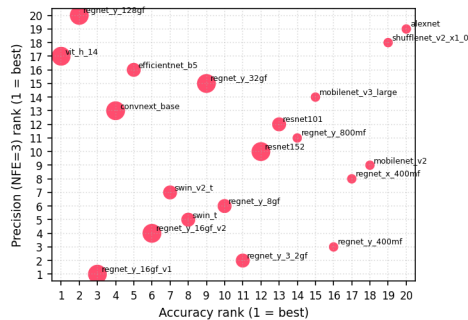
Method	3	4	5	6	7	8	9
FID (\downarrow)							
DDIM (Song et al., 2021a)	45.05	27.72	23.93	23.06	22.99	22.96	22.97
DPM-Solver++ (Lu et al., 2022b)	45.33	26.12	22.56	22.48	22.79	22.90	23.11
BNS-Solver (Shaul et al., 2024)	48.16	26.37	21.04	20.66	20.79	21.13	21.64
DS-Solver (Wang et al., 2025)	48.65	29.15	21.66	20.65	21.43	21.80	22.27
Dual-Solver (Ours)	21.79	19.40	18.81	18.52	19.57	19.43	19.77
CLIP (\uparrow)							
DDIM (Song et al., 2021a)	0.4656	0.4782	0.4813	0.4821	0.4824	0.4824	0.4824
DPM-Solver++ (Lu et al., 2022b)	0.4652	0.4779	0.4813	0.4821	0.4822	0.4820	0.4820
BNS-Solver (Shaul et al., 2024)	0.4651	0.4772	0.4800	0.4807	0.4808	0.4809	0.4808
DS-Solver (Wang et al., 2025)	0.4635	0.4765	0.4808	0.4816	0.4818	0.4818	0.4816
Dual-Solver (Ours)	0.4795	0.4801	0.4821	0.4837	0.4843	0.4844	0.4849

E CLASSIFIER ACCURACY VS. SAMPLE QUALITY

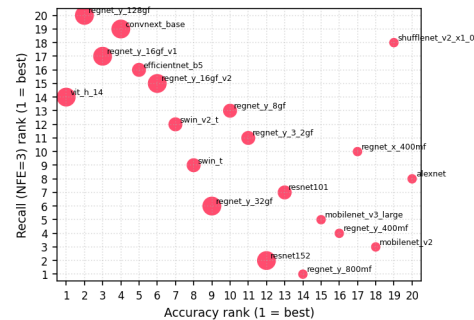
As examined in Sec. 6.2, we study the relationship between classifier accuracy and FID. In this section, Table 10 provides detailed numerical results, and Fig. 10 analyzes the trend from the perspectives of precision and recall (Kynkäänniemi et al., 2019). Table 10 reports the pretrained weights from TorchVision (maintainers & contributors, 2016) and the results of training Dual-Solver with each set of weights. Using a GM-DiT (Chen et al., 2025) backbone, we present FID, precision, and recall at NFE= 3 and 9. The best value for each metric is highlighted in bold.

Table 10: **ImageNet per-classifier metrics.** Top-5 accuracy, FID, precision/recall at NFE=3 and 9, and GFLOPs; FID is measured on 10k samples after training Dual-Solver for each classifier.

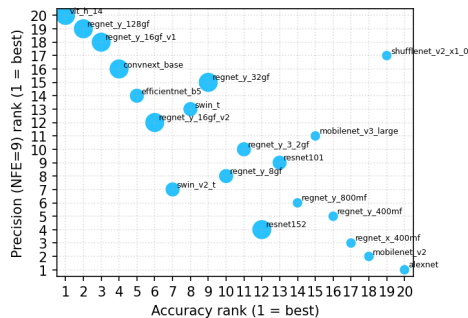
Weights ³	Top-5 Acc. (%)	FID@3	FID@9	Precision@3	Precision@9	Recall@3	Recall@9	GFLOPs
ViT_H.14.Weights.IMAGENET1K_SWAG_E2E_V1	98.694	10.71	6.72	0.8376	0.8960	0.7420	0.7311	1016.72
RegNet_Y.128GF.Weights.IMAGENET1K_SWAG_LINEAR_V1	97.844	12.99	5.61	0.8230	0.8996	0.7327	0.7336	127.52
RegNet_Y.16GF.Weights.IMAGENET1K_SWAG_LINEAR_V1	97.244	9.97	5.32	0.8573	0.9007	0.7392	0.7467	15.91
ConvNext_Base.Weights.IMAGENET1K_V1	96.870	10.28	5.78	0.8529	0.9040	0.7359	0.7327	15.36
EfficientNet_B5.Weights.IMAGENET1K_V1	96.628	11.53	6.01	0.8379	0.9048	0.7393	0.7310	10.27
RegNet_Y.16GF.Weights.IMAGENET1K_V2	96.328	9.63	5.21	0.8559	0.9061	0.7396	0.7455	15.91
SwinV2_T.Weights.IMAGENET1K_V1	96.132	9.73	5.34	0.8555	0.9078	0.7439	0.7410	5.94
Swin_T.Weights.IMAGENET1K_V1	95.776	9.65	5.29	0.8559	0.9055	0.7448	0.7425	4.49
RegNet_Y.32GF.Weights.IMAGENET1K_V1	95.340	9.55	5.04	0.8520	0.9043	0.7464	0.7455	32.28
RegNet_Y.8GF.Weights.IMAGENET1K_V1	95.048	9.54	5.10	0.8559	0.9074	0.7438	0.7414	8.47
RegNet_Y.3.2GF.Weights.IMAGENET1K_V1	94.576	9.50	5.01	0.8564	0.9065	0.7448	0.7436	3.18
ResNet152.Weights.IMAGENET1K_V1	94.046	9.49	5.03	0.8537	0.9089	0.7493	0.7459	11.51
ResNet101.Weights.IMAGENET1K_V1	93.546	9.59	4.98	0.8529	0.9067	0.7459	0.7454	7.80
RegNet_Y.80MF.Weights.IMAGENET1K_V1	93.136	9.40	4.99	0.8532	0.9081	0.7508	0.7437	0.83
MobileNet_V3_Large.Weights.IMAGENET1K_V2	92.566	9.44	4.87	0.8523	0.9062	0.7467	0.7475	0.22
RegNet_Y.40MF.Weights.IMAGENET1K_V1	91.716	9.50	4.87	0.8560	0.9082	0.7470	0.7489	0.40
RegNet_X.40MF.Weights.IMAGENET1K_V1	90.950	9.60	4.93	0.8553	0.9091	0.7448	0.7410	0.41
MobileNet_V2.Weights.IMAGENET1K_V1	90.286	9.52	5.03	0.8541	0.9118	0.7471	0.7331	0.30
ShuffleNet_V2_X1.0.Weights.IMAGENET1K_V1	88.316	11.57	5.51	0.8368	0.9017	0.7388	0.7436	0.14
AlexNet.Weights.IMAGENET1K_V1	79.066	11.20	6.22	0.8300	0.9163	0.7455	0.7187	0.71



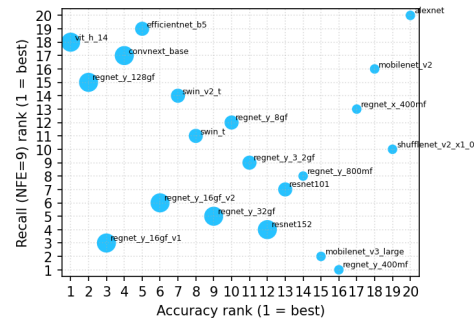
(a) Accuracy vs. Precision @ NFE= 3



(b) Accuracy vs. Recall @ NFE= 3



(c) Accuracy vs. Precision @ NFE= 9



(d) Accuracy vs. Recall @ NFE= 9

Figure 10: **Accuracy vs. precision/recall.** We select 20 TorchVision classifiers sorted by accuracy, learn the Dual-Solver for each, and report precision/recall at NFE=3 and 9 on 10k samples.

Classifier accuracy vs. precision and recall. Fig. 10 (a) and (b) plot precision and recall versus classifier accuracy at NFE= 3, respectively. The results show little relationship between accuracy

³<https://docs.pytorch.org/vision/main/models.html>

and precision, while accuracy versus recall follows the V-shape seen in Fig. 6. In other words, neither very high nor very low accuracy helps recall; a moderate level yields higher recall. The pattern differs at NFE= 9. Fig. 10 (c) and (d) plot precision and recall versus classifier accuracy at NFE= 9. Unlike the NFE= 3 case, precision exhibits a strong negative correlation with accuracy, and accuracy appears largely unrelated to recall.

OpenCLIP accuracy vs. FID. Table 11 reports the FID evaluation used to select the CLIP model for learning Dual-Solver on the text-to-image task. All weights are from OpenCLIP (Ilharco et al., 2021) and are available from the official repository⁴. Using the SANA (Xie et al., 2024) backbone we trained the Dual-Solver for 20k steps. At NFE = 3 and 6, we generated 30k samples with MSCOCO 2014 (Lin et al., 2014) prompts and measured FID on the evaluation set. Based on these results, we chose the RN101 weights—which achieved an FID of 18.52 at NFE = 6—as the model for learning the Dual-Solver in the main text-to-image experiment. Notably, this result also indicates that models with somewhat lower classification accuracy can yield lower FID.

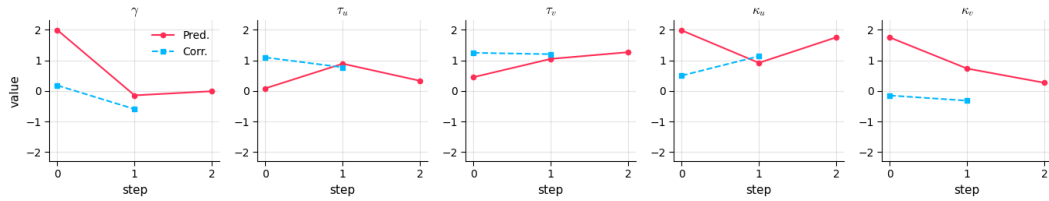
Table 11: **OpenCLIP per-classifier metrics.** MSCOCO accuracy, FID at NFE = 3 and 6, and GFLOPs; FID is measured on 30k samples after learning Dual-Solver for each classifier.

Weights	MSCOCO Acc. (%)	FID@3	FID@6	GFLOPs
ViT-H-14-378-quickgelu, dfn5b	63.76	23.98	23.28	1054.05
coca_ViT-L-14, mscoco_finetuned_laion2b_s13b_b90k	60.28	23.09	21.22	214.52
EVA02-E-14, laion2b_s4b_b115k	58.92	22.41	21.12	1007.93
convnext_xxlarge, laion2b_s34b_b82k_augreg	58.34	21.05	20.86	800.88
ViT-B-16-SigLIP-256, webli	57.24	21.03	19.87	57.84
EVA02-L-14-336, merged2b_s6b_b61k	56.05	23.15	23.28	167.50
ViT-L-14, commonpool_xl_laion_s13b_b90k	55.13	23.46	22.81	175.33
convnext_base_w, laion_aesthetic_s13b_b82k	52.38	20.97	19.86	49.38
convnext_base_w_320, laion_aesthetic_s13b_b82k_augreg	51.42	20.69	20.26	175.33
ViT-B-16-plus-240, laion400m_e32	49.79	21.66	21.18	64.03
ViT-B-32, laion2b_e16	47.68	23.75	23.49	14.78
ViT-B-32-quickgelu, metaclip_fullcc	46.62	22.46	21.42	14.78
RN50x16, openai	45.38	22.49	21.36	33.34
ViT-B-32, laion400m_e31	43.27	22.40	21.70	14.78
RN101, openai	40.25	21.78	18.52	25.50
ViT-B-16, commonpool_l_text_s1b_b8k	37.30	23.50	23.54	41.09
ViT-B-16, commonpool_l_s1b_b8k	28.55	22.94	24.73	41.09
ViT-B-32, commonpool_m_text_s128m_b4k	14.52	22.39	22.55	14.78
ViT-B-32, commonpool_s_clip_s13m_b4k	2.24	22.32	21.31	14.78
coca_ViT-B-32, mscoco_finetuned_laion2b_s13b_b90k	0.60	23.76	23.14	33.34

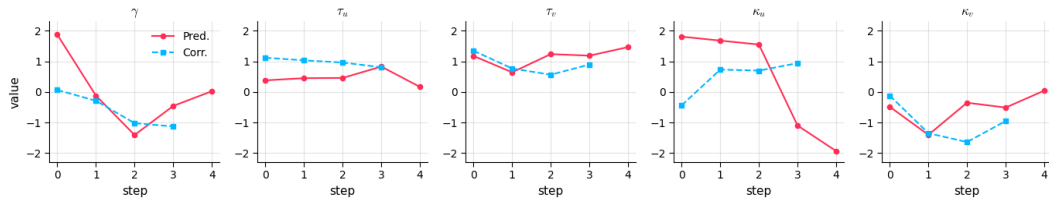
⁴https://github.com/mlfoundations/open_clip

F LEARNED PARAMETERS

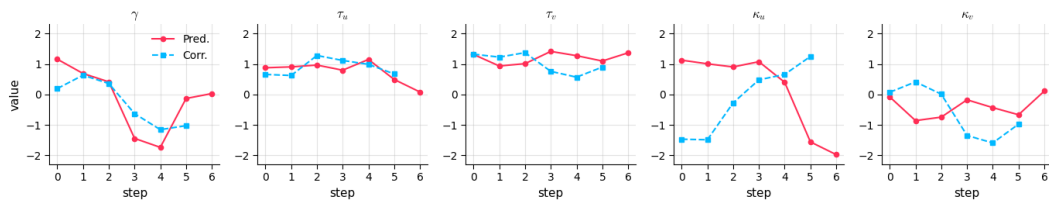
In Figs. 11, 12, 13, 14 we plot the Dual-Solver parameters learned via classification-based learning (Sec. 5.2), as defined in Sec. 4.2. For the DiT, GM-DiT, SANA, and PixArt- α backbones, we set CFG to 1.5, 1.4, 4.5, and 3.5, respectively, and train for 20k steps. We plot results at NFE = 3, 5, 7, and 9 for DiT and GM-DiT, and at NFE = 3, 4, 5, and 6 for SANA and PixArt- α . Separate parameter sets are learned for the predictor and the corrector, and they are shown in different colors. Within the same backbone, the parameter curves are similar even across different NFEs. We conjecture that this arises from the backbone’s intrinsic trajectory.



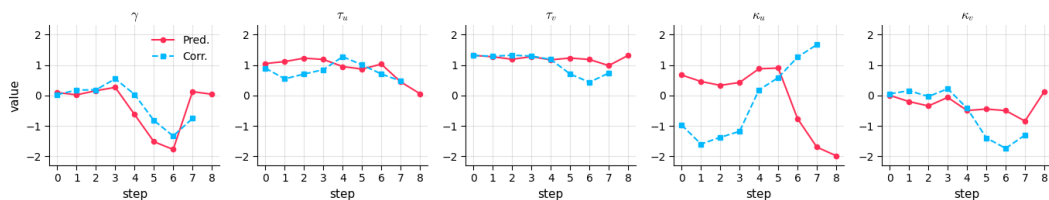
(a) NFE = 3



(b) NFE = 5

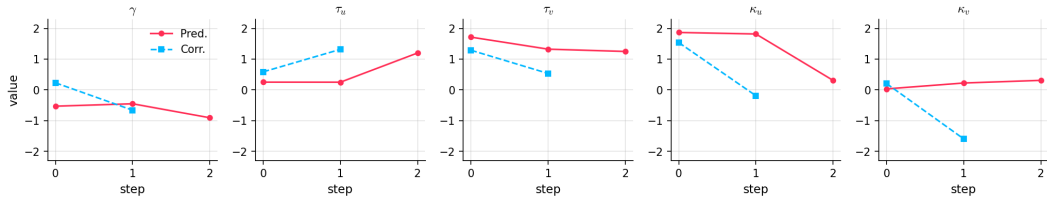


(c) NFE = 7

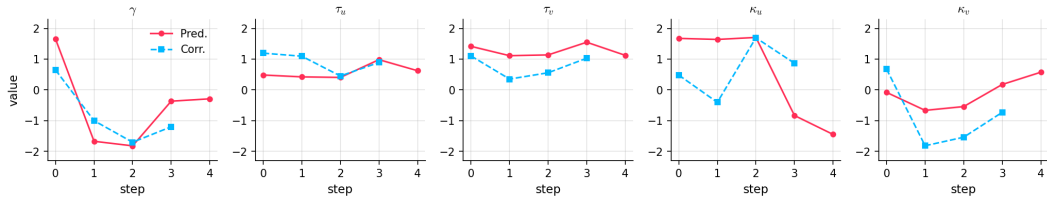


(d) NFE = 9

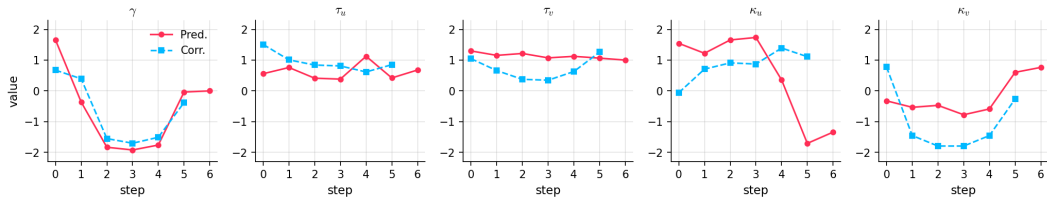
Figure 11: **Learned parameters.** $\{\gamma, \tau_u, \tau_v, \kappa_u, \kappa_v\}$ for DiT (Peebles & Xie, 2023).



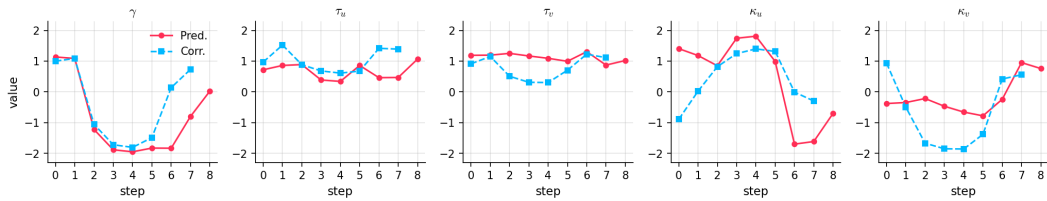
(a) NFE = 3



(b) NFE = 5

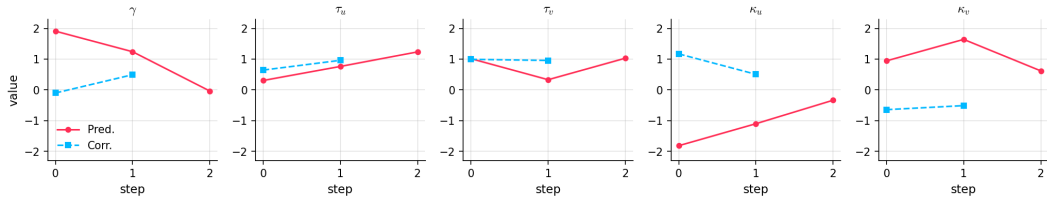


(c) NFE = 7

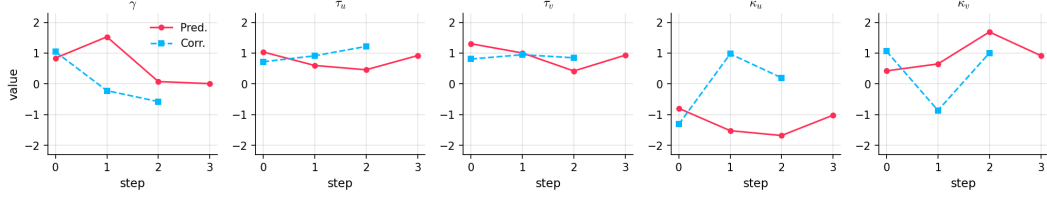


(d) NFE = 9

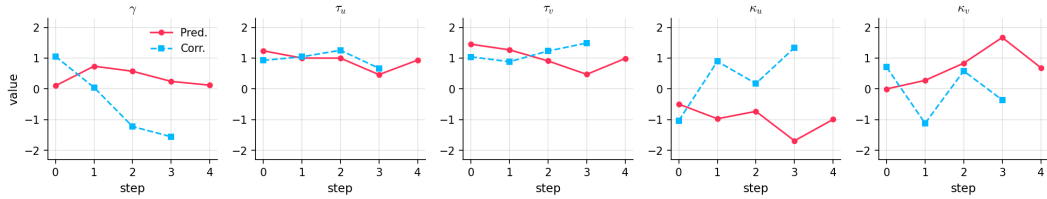
Figure 12: Learned parameters. $\{\gamma, \tau_u, \tau_v, \kappa_u, \kappa_v\}$ for GM-DiT (Chen et al., 2025).



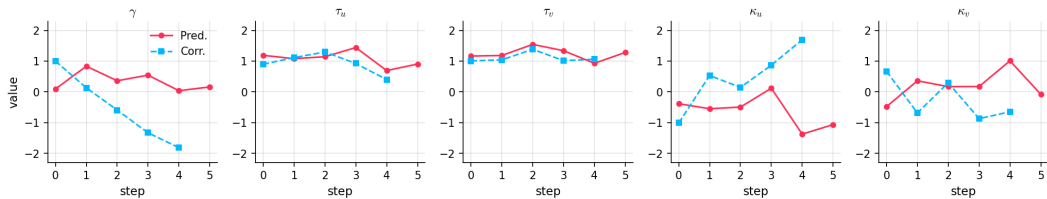
(a) NFE = 3



(b) NFE = 4

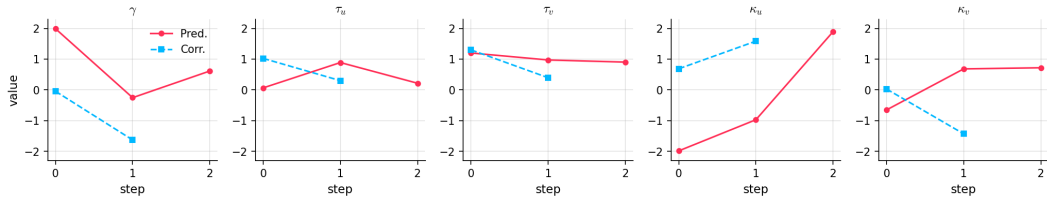


(c) NFE = 5

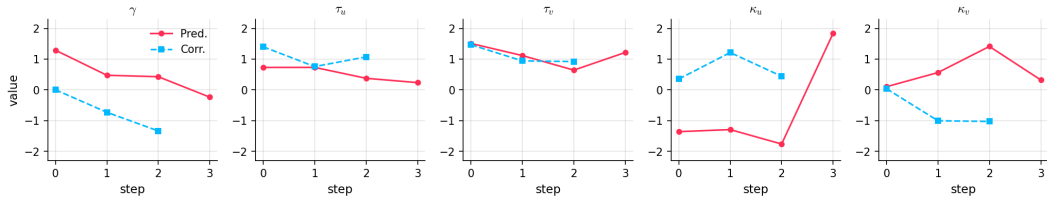


(d) NFE = 6

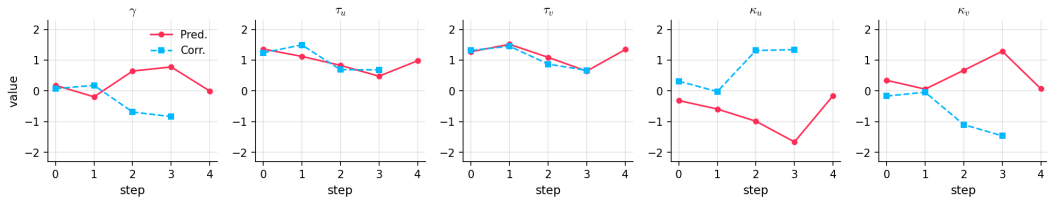
Figure 13: Learned parameters. $\{\gamma, \tau_u, \tau_v, \kappa_u, \kappa_v\}$ for SANA (Xie et al., 2024).



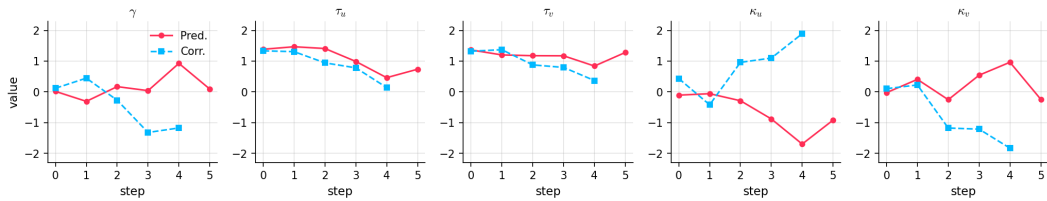
(a) NFE = 3



(b) NFE = 4



(c) NFE = 5



(d) NFE = 6

Figure 14: **Learned parameters.** $\{\gamma, \tau_u, \tau_v, \kappa_u, \kappa_v\}$ for PixArt- α (Chen et al., 2023).

G DETAILS OF PARAMETER INTERPOLATION ACROSS NFES

In this section, we describe how to interpolate the parameters discussed in Sec. 6.3 so that they can be used at other NFES. The parameters of Dual-Solver,

$$\phi = \{\gamma^{\text{pred}}, \tau_u^{\text{pred}}, \tau_v^{\text{pred}}, \kappa_u^{\text{pred}}, \kappa_v^{\text{pred}}, \gamma^{\text{corr}}, \tau_u^{\text{corr}}, \tau_v^{\text{corr}}, \kappa_u^{\text{corr}}, \kappa_v^{\text{corr}}\},$$

are given as arrays whose length equals the NFE. (The corrector parameters have length $\text{NFE}-1$, and we match the length to NFE by repeating the last element.) For example, $\gamma^{\text{pred}} = (\gamma_0^{\text{pred}}, \dots, \gamma_{\text{NFE}-1}^{\text{pred}})$. To interpolate these parameters, we consider the following linear interpolation scheme for a generic array.

Definition G.1 (Linear interpolation). Let $f^{(M)} = (f_0^{(M)}, \dots, f_{M-1}^{(M)})$ be an array of length M . The linearly interpolated array $\text{Interp}(f^{(M)}; N)$ of length N is defined as follows. First, set

$$t_i = \frac{i}{N-1} (M-1), \quad j_i = \lfloor t_i \rfloor, \quad \alpha_i = t_i - j_i,$$

and for each $i = 0, \dots, N-1$ define

$$\text{Interp}(f^{(M)}; N)[i] = (1 - \alpha_i) f_{j_i}^{(M)} + \alpha_i f_{j_i+1}^{(M)}.$$

Definition G.2 (Averaged linear interpolation). Let $M < N < L$ be three NFES, and let $f^{(M)} \in \mathbb{R}^M$ and $f^{(L)} \in \mathbb{R}^L$ be the corresponding arrays. We first obtain their linearly interpolated versions of length N ,

$$\tilde{f}^{(M)} = \text{Interp}(f^{(M)}; N), \quad \tilde{f}^{(L)} = \text{Interp}(f^{(L)}; N).$$

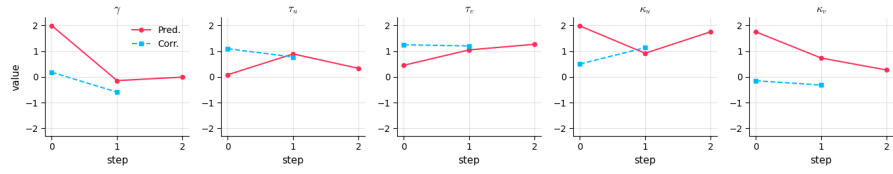
We then use the relative position of N between M and L as weights and define

$$w_M = \frac{L-N}{L-M}, \quad w_L = \frac{N-M}{L-M},$$

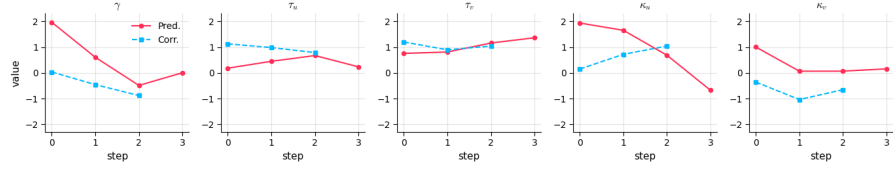
and, for each $i = 0, \dots, N-1$,

$$\text{Interp}(f^{(M)}, f^{(L)}; N)[i] = w_M \tilde{f}^{(M)}[i] + w_L \tilde{f}^{(L)}[i].$$

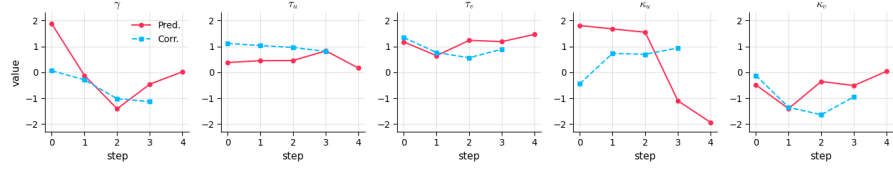
Using this procedure, we obtain the array for an intermediate NFE from the two arrays at neighboring NFES, and apply the same construction to every parameter in ϕ . Examples of interpolated parameters are shown in Fig. 15. Specifically, we obtain the parameters for $\text{NFE} = 4, 6$, and 8 by interpolating those learned at $\text{NFE} = (3, 5), (5, 7)$, and $(7, 9)$, respectively.



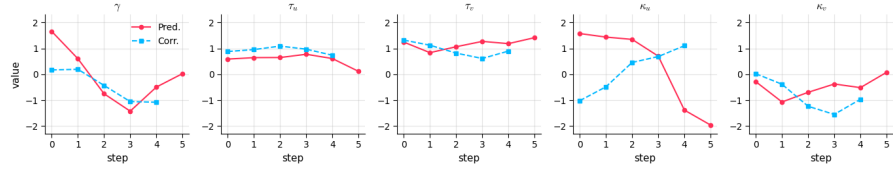
(a) NFE = 3



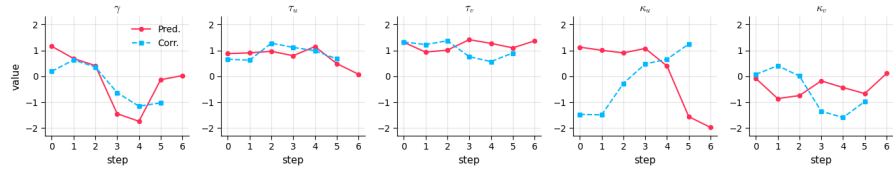
(b) NFE = 4, interpolated from (3, 5)



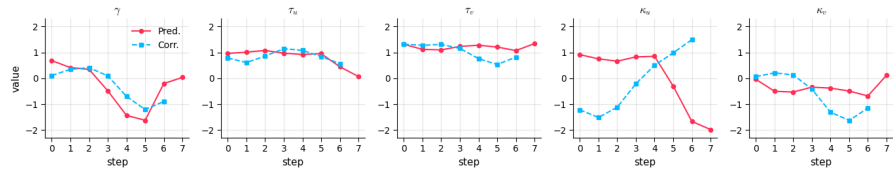
(c) NFE = 5



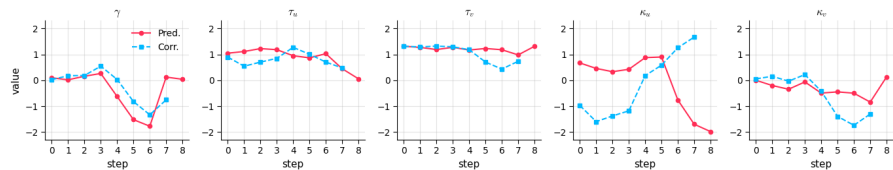
(d) NFE = 6, interpolated from (5, 7)



(e) NFE = 7

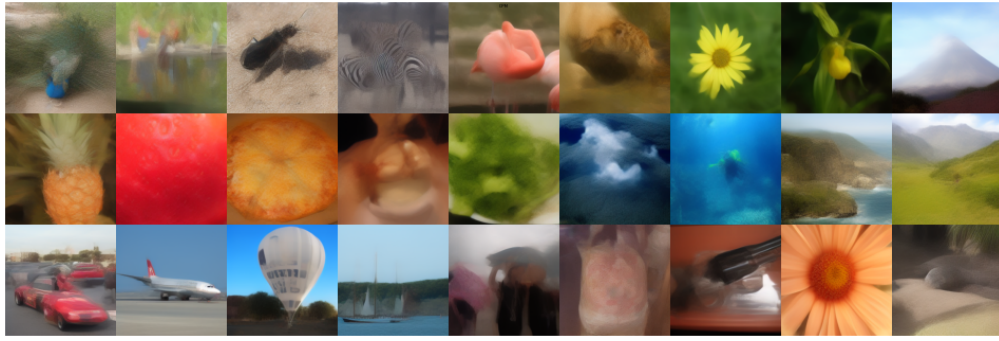


(f) NFE = 8, interpolated from (7, 9)



(g) NFE = 9

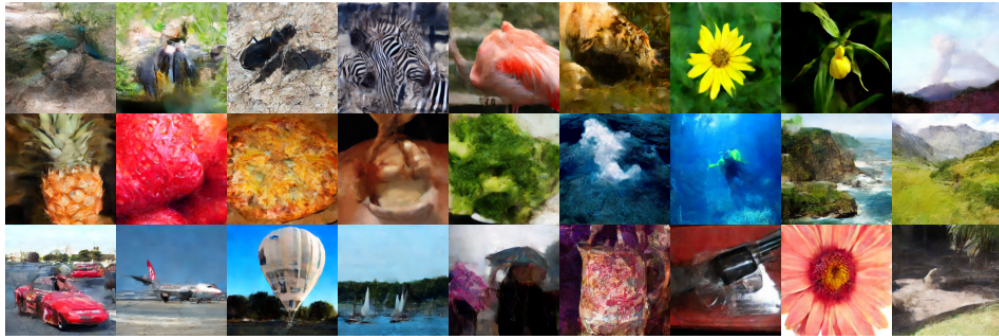
Figure 15: Interpolated parameters. $\{\gamma, \tau_u, \tau_v, \kappa_u, \kappa_v\}$ for DiT (Peebles & Xie, 2023).



(a) DPM-Solver++ (Lu et al., 2022b)



(b) Dual-Solver (Ours)



(c) BNS-Solver (Shaul et al., 2024)



(d) DS-Solver (Wang et al., 2025)

Figure 16: **Additional sampling results.** DiT-XL/2 256×256 (NFE=4, CFG=1.5)



(a) DPM-Solver++ (Lu et al., 2022b)



(b) Dual-Solver (Ours)



(c) BNS-Solver (Shaul et al., 2024)



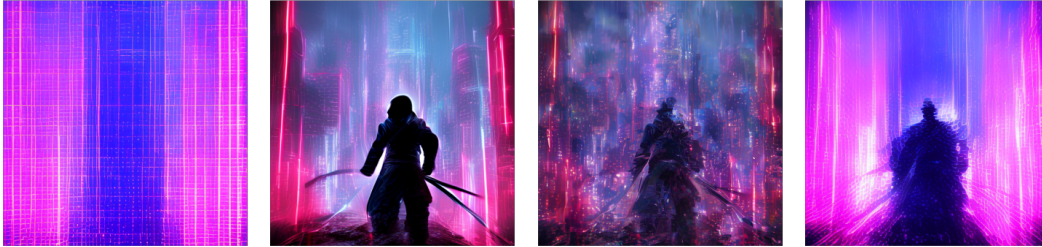
(d) DS-Solver (Wang et al., 2025)

Figure 17: **Additional sampling results.** GM-DiT 256×256 (NFE=3, CFG=1.4)

Golden autumn park of falling leaves, graceful girl playing violin, flowing satin dress, impressionist brush strokes



Towering neon-lit cyberpunk skyline, armored samurai with glowing katana, chrome textures, cinematic digital art



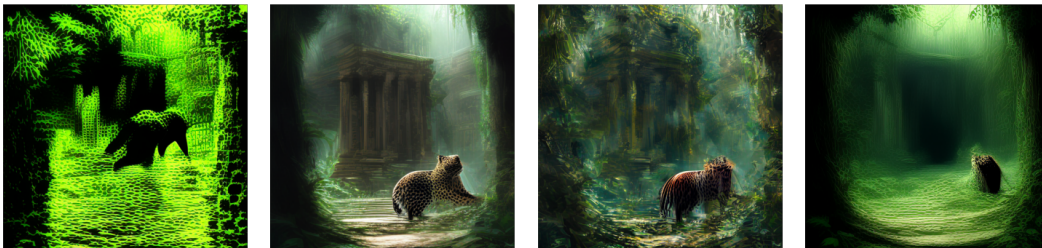
Remote desert oasis at sunrise, powerful white stallion galloping, glossy mane and muscles, hyperreal photo realism



Vibrant tropical beach at sunset, pirate ship anchored offshore, weathered wood hull, playful cartoon illustration



Overgrown jungle temple ruins, spotted leopard stalking prey, sleek fur patterns, painterly realism concept art



(a) DPM-Solver++
(Lu et al., 2022b)

(b) Dual-Solver (Ours)

(c) BNS-Solver
(Shaul et al., 2024)

(d) DS-Solver
(Wang et al., 2025)

Figure 18: **Additional sampling results.** SANA (Xie et al., 2024), NFE=3, CFG=4.5.

In a misty emerald forest clearing, a majestic golden wolf with silky glowing fur, painted in luminous oil style



Towering neon-lit cyberpunk skyline, armored samurai with glowing katana, chrome textures, cinematic digital art



Vast desert beneath a starry cosmos, colossal sphinx carved in sandstone, rough surface, surreal dreamlike painting



Sleek futuristic laboratory interior, humanoid AI robot in motion, polished steel surfaces, sci-fi blueprint style



Storm-drenched battlefield ruins, giant mech rising from fire, rusted armor plates, gritty photorealistic concept art



(a) DPM-Solver++
(Lu et al., 2022b)

(b) Dual-Solver (Ours)

(c) BNS-Solver
(Shaul et al., 2024)

(d) DS-Solver
(Wang et al., 2025)

Figure 19: **Additional sampling results.** PixArt- α (Chen et al., 2023), NFE=5, CFG=3.5.