Image Diffusion Preview with Consistency Solver

Supplementary Material

Fu-Yun Wang[1,2], Hao Zhou[1], Liangzhe Yuan[1], Sanghyun Woo[1], Boqing Gong[1], Bohyung Han[1,3], Ming-Hsuan Yang[1], Han Zhang[1], Yukun Zhu[1], Ting Liu[1], Long Zhao[1]

[1]Google DeepMind [2]The Chinese University of Hong Kong [3]Seoul National University

A. Common diffusion ODE solvers via Taylor expansion

The exact solution of Eq. (3) requires numerical approximation of

$$\Delta \mathbf{y}_{t\to s} = \int_{n_t}^{n_s} \boldsymbol{\epsilon}(\mathbf{x}_{t_n}, t_n)\, \mathrm{d}n. \tag{S1}$$

Let $h = n_s - n_t$. The Taylor expansion of the integrand around n_t yields

$$\int_{n_t}^{n_s} \boldsymbol{\epsilon}(\mathbf{x}_{t_n}, t_n)\, \mathrm{d}n = h\, \boldsymbol{\epsilon}(\mathbf{x}_t, t) + \frac{h^2}{2} \frac{\mathrm{d}}{\mathrm{d}n} \boldsymbol{\epsilon}(\mathbf{x}_{t_n}, t_n)\Big|_{n_t} + \frac{h^3}{6} \frac{\mathrm{d}^2}{\mathrm{d}n^2} \boldsymbol{\epsilon}(\mathbf{x}_{t_n}, t_n)\Big|_{n_t} + \cdots . \tag{S2}$$

For brevity, we denote

$$\boldsymbol{\epsilon}_t \triangleq \boldsymbol{\epsilon}(\mathbf{x}_t, t), \tag{S3}$$

and similarly for other time points (*e.g.*, s).

A.1. First-order: DDIM / Euler (naïve)

$$\Delta \mathbf{y}_{t\to s} \approx h\, \boldsymbol{\epsilon}_t\, . \tag{S4}$$

Retains only the zeroth-order term in Eq. (S2).

A.2. Second-order: DPM-Solver-2 / midpoint

The midpoint method uses one evaluation near the interval center:

$$\Delta \mathbf{y}_{t\to s} \approx h\, \boldsymbol{\epsilon}_r, \quad n_r \approx n_t + \frac{h}{2}\, . \tag{S5}$$

To see second-order accuracy, approximate the missing derivative with a centered finite difference:

$$\frac{\mathrm{d}}{\mathrm{d}n} \boldsymbol{\epsilon}\Big|_{n_t} \approx \frac{\boldsymbol{\epsilon}_r - \boldsymbol{\epsilon}_t}{h/2}. \tag{S6}$$

Insert into the desired second-order truncation:

$$h\, \boldsymbol{\epsilon}_t + \frac{h^2}{2} \cdot \frac{\boldsymbol{\epsilon}_r - \boldsymbol{\epsilon}_t}{h/2} = h\, \boldsymbol{\epsilon}_t + h\left(\boldsymbol{\epsilon}_r - \boldsymbol{\epsilon}_t\right) = h\, \boldsymbol{\epsilon}_r. \tag{S7}$$

Thus $h\, \boldsymbol{\epsilon}_r$ exactly matches the second-order Taylor integral when the first derivative is estimated by a midpoint difference. DPM-Solver-2 exploits this insight, typically choosing $n_r = \sqrt{n_t n_s}$ (geometric midpoint in noise-scale space).

B. Common diffusion ODE solvers interpreted using *ConsistencySolver*

ConsistencySolver treats the coefficients in Eq. (5) as learnable unknowns. Here we show that several widely adopted diffusion solvers [23, 24, 38] can be easily interpreted using the form of *ConsistencySolver*.

For notational simplicity, we denote $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t_i}, t_i)$ simply as $\boldsymbol{\epsilon}_i$ throughout this section.

DDIM (naive approximation) performs the update:

$$\mathbf{y}_{t_{i+1}} = \mathbf{y}_{t_i} + (n_{t_{i+1}} - n_{t_i})\boldsymbol{\epsilon}_i\, . \tag{S8}$$

Comparing with Eq. (5), we can have the naive approximation corresponds to a one-step method ($m = 1$) with the coefficient $w_1 = 1$.

PNDM utilizes the explicit 4-step Adams-Bashforth method [34]. For the Ininial Value Problem (IVP) $\mathrm{d}\mathbf{y}/\mathrm{d}n = \boldsymbol{\epsilon}$, the update is:

$$\mathbf{y}_{t_{i+1}} = \mathbf{y}_{t_i} + \frac{\Delta n_i}{24}\left[55\boldsymbol{\epsilon}_i - 59\boldsymbol{\epsilon}_{i-1} + 37\boldsymbol{\epsilon}_{i-2} - 9\boldsymbol{\epsilon}_{i-3}\right], \tag{S9}$$

where $\Delta n_i = n_{t_{i+1}} - n_{t_i}$. This corresponds to $m = 4$ with coefficients:

$$w_1 = \frac{55}{24}, \quad w_2 = -\frac{59}{24}, \quad w_3 = \frac{37}{24}, \quad w_4 = -\frac{9}{24}, \tag{S10}$$

of the proposed the *ConsistencySolver* defined in Eq. (5).

DPM-Solver-2 (midpoint approximation) uses an evaluation at an intermediate point t_i (corresponding to $n_{t_i} = \sqrt{n_{t_{i-1}} n_{t_{i+1}}}$):

$$\begin{aligned}
\mathbf{y}_{t_i} &= \mathbf{y}_{t_{i-1}} + (n_{t_i} - n_{t_{i-1}})\boldsymbol{\epsilon}_{i-1}\, , \\
\mathbf{y}_{t_{i+1}} &= \mathbf{y}_{t_{i-1}} + (n_{t_{i+1}} - n_{t_{i-1}})\boldsymbol{\epsilon}_i \\
&= \mathbf{y}_{t_i} + (n_{t_{i+1}} - n_{t_{i-1}})\boldsymbol{\epsilon}_i - (n_{t_i} - n_{t_{i-1}})\boldsymbol{\epsilon}_{i-1} \\
&= \mathbf{y}_{t_i} + (n_{t_{i+1}} - n_{t_i})\Big[\frac{(n_{t_{i+1}} - n_{t_{i-1}})}{(n_{t_{i+1}} - n_{t_i})}\boldsymbol{\epsilon}_i \\
&\quad - \frac{(n_{t_i} - n_{t_{i-1}})}{(n_{t_{i+1}} - n_{t_i})}\boldsymbol{\epsilon}_{i-1}\Big]
\end{aligned} \tag{S11}$$

Comparing with Eq. (5), we can have DPM-Solver-2 corresponds to two-stages computation. When i is even (*i.e.*, $0, 2, 4, \ldots$), the approximation corresponds to a one-step

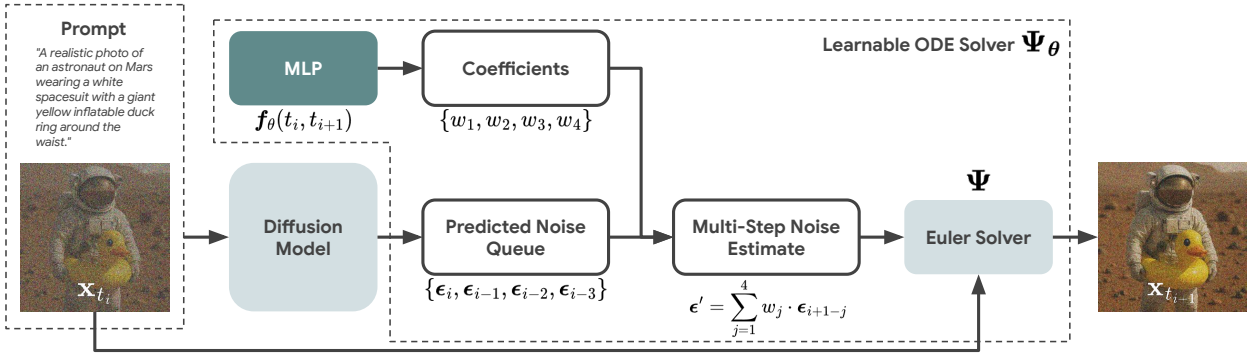

Figure S1. Workflow of the generalized learnable ODE solver $\boldsymbol{\Psi}_{\boldsymbol{\theta}}$ with Order 4 ($m = 4$). At each sampling step, the diffusion model predicts noise ϵ_i conditioned on the input prompt and timestep. A learnable neural network $\boldsymbol{f}_\theta$ generates adaptive coefficients w_j, $j = 1, 2, 3, 4$ from current timestep t_i, and target timestep t_{i+1} , which are used to form a multi-step noise estimate $\boldsymbol{\epsilon}' = \sum_{j=1}^{4} w_j \cdot \boldsymbol{\epsilon}_{i+1-j}$. The ODE solver $\boldsymbol{\Psi}_{\boldsymbol{\theta}}$ then updates the sample from $\mathbf{x}_{t_i}$ to $\mathbf{x}_{t_{i+1}}$. This approach enables more accurate and stable integration in the generative sampling process.

method ($m = 1$) with the coefficient $w_1 = 1$. When i is odd, the approximation corresponds to a two-step method ($m = 2$) with the coefficient $w_1 = \frac{(n_{t_{i+1}} - n_{t_{i-1}})}{(n_{t_{i+1}} - n_{t_i})}, w_2 = -\frac{(n_{t_i} - n_{t_{i-1}})}{(n_{t_{i+1}} - n_{t_i})}$.

C. Visualization of *ConsistencySolver*

We visualize the computation paradigm of the proposed *ConsistencySolver* in Fig. S1, taking Order 4 ($m = 4$) as an example.

D. Implementation details

D.1. ConsistencySolver training

Training dataset. We randomly sample 2,000 prompts from the LAION dataset [35] and generate corresponding images using a 40-step multistep DPM-Solver, forming noise-prompt-target image triplets as our training data.

Training procedure. All experiments are conducted on a single H100 GPU. For each training iteration, we select one prompt-noise pair and replicate it 80 times. We then apply the trainable *ConsistencySolver* to generate 80 different sampling trajectories with random perturbations. Following the PPO algorithm, we increase the probability of high-reward trajectories while suppressing low-reward ones. By default, we use Order-4 solver configurations. The MLP network in *ConsistencySolver* is trained from scratch using a learning rate of 1×10^{-4} for 3,000 iterations, requiring approximately 12 H100 GPU hours in total.

D.2. Distillation baseline training

Beyond the proposed RL-based training approach, we explore distillation-based alternatives to optimize the dynamic coefficients in *ConsistencySolver*. We investigate two distillation schemes:

Final-state distillation. This approach treats the entire few-step diffusion sampling chain as differentiable and directly uses the negative reward at the final state as the loss function. Gradients are backpropagated through the complete inference chain to optimize the parameters. While conceptually straightforward, this method exhibits significant drawbacks. First, backpropagating through the entire chain requires computing gradients not only for the *ConsistencySolver* MLP but also for the underlying diffusion model (typically containing billions of parameters), substantially increasing computational cost. Second, we observe severe training instability, with the MLP failing to converge effectively in practice.

Trajectory distillation. Inspired by prior work [44, 54], we propose a trajectory-based distillation method, referred to as *Ours-Distill* in the main text. This approach requires storing the complete 40-step trajectory from the multistep DPM-Solver (introducing additional storage overhead). The objective is to match each intermediate state in the few-step *ConsistencySolver* sampling to corresponding states in the 40-step reference trajectory. For example, when performing 5-step sampling, each *ConsistencySolver* step should align with 8 steps of the reference solver. We use the negative similarity between these states as the loss function for backpropagation. This method significantly outperforms final-state distillation but still falls short of the RL-based approach, as demonstrated in our quantitative comparisons in Tab. 2.

Training dataset. We use the same 2,000 training samples as for *ConsistencySolver* training to ensure fair comparison.

Table S1. Ablation study on model structure at 8 and 10 steps. Best results per metric in **bold**.

Model	Steps	Dep.↑	Inc.↑	Img.↑	CLIP↑	DINO↑
8 Steps						
Hidden Dim 32	8	22.08	91.55	23.07	96.24	90.96
Hidden Dim 256	8	**22.22**	**91.68**	**23.56**	**96.36**	**91.14**
Hidden Dim 1024	8	21.82	91.30	22.36	96.04	90.57
Deep (12-Layer MLP)	8	22.00	91.20	22.60	96.14	90.68
10 Steps						
Hidden Dim 32	10	24.68	93.67	24.80	97.16	93.31
Hidden Dim 256	10	**25.01**	**93.85**	**25.57**	**97.30**	**93.67**
Hidden Dim 1024	10	24.12	93.23	23.96	96.92	92.67
Deep (12-Layer MLP)	10	24.38	93.39	24.22	96.99	93.12

D.3. Preview study experimental protocol

Evaluation datasets. For the preview study, we evaluate on three datasets: (1) GenEval evaluation set containing 553 prompts [11], (2) COCO 2017 validation set with 5,000 prompts [22], and (3) 5,000 randomly sampled prompts from LAION [35].

Evaluation with LLM. We use Claude Sonnet 4 as an automated judge to simulate a discerning user. The system prompt is designed to enforce strict evaluation criteria:

> *"You are a very picky user evaluating an AI-generated image for the prompt '{prompt}'. Be extremely critical—only approve if it perfectly matches the description in composition, quality, details, and realism. Respond with ONLY 'SATISFIED' if it's perfect, or 'NOT_SATISFIED: [brief reason]' otherwise. Keep the reason under 50 words."*

This ensures the LLM judges each generated image with high standards, accepting only those that closely align with the prompt requirements.

Human evaluation. To complement LLM evaluation, we conduct human studies with real users. For each prompt, we pre-generate 10 images and record their generation times. These images are organized into questionnaires where participants sequentially evaluate whether each image satisfies the prompt. Participants stop at the first satisfactory image; if all images are unsatisfactory, the trial is discarded as discussed in the main text. We recruit 20 volunteers, each responsible for evaluating 100 prompts uniformly sampled across all test datasets, resulting in comprehensive human feedback on the practical effectiveness of our preview mechanism.

D.4. Ablation study on model structures

We analyze architectural variants of *ConsistencySolver*, varying hidden dimension size and testing a deep 12-layer MLP with residual LayerNorm, evaluated at 8 and 10 steps.

Table S2. Ablation study on solver order at 5, 8, and 10 steps. Best results per metric in **bold**.

Orders	Steps	Dep.↑	Inc.↑	Seg.↑	Img.↑	CLIP↑	DINO↑
5 Steps							
Order 2	5	**19.33**	**87.30**	69.36	**20.84**	94.40	86.39
Order 3	5	19.15	86.46	68.93	20.26	93.80	85.83
Order 4	5	19.29	87.07	**69.42**	20.75	94.22	86.35
Order 5	5	**19.33**	87.16	69.38	20.64	**94.33**	**86.44**
8 Steps							
Order 2	8	22.12	91.59	78.56	23.34	96.31	91.03
Order 3	8	22.14	91.57	77.92	23.20	96.26	90.81
Order 4	8	**22.15**	**91.65**	**78.52**	**23.43**	**96.35**	**91.09**
Order 5	8	22.12	91.65	78.19	23.15	96.33	90.97
10 Steps							
Order 2	10	24.72	93.74	82.86	25.16	**97.25**	93.45
Order 3	10	24.66	93.74	82.68	25.23	97.23	93.29
Order 4	10	**24.94**	**93.88**	**83.22**	**25.32**	**97.25**	**93.48**
Order 5	10	24.72	93.79	82.78	24.88	97.18	93.36

According to Tab. S1, the 256-dimensional model consistently outperforms others, delivering superior results in image similarity, semantic alignment, and overall consistency. Larger dimensions (*e.g.*, 1024) slightly enhance depth estimation but compromise balance and efficiency. The deep MLP variant shows no meaningful advantage over the standard 256-dim architecture, suggesting that moderate capacity is sufficient for the task.

D.5. Ablation study on solver orders

We assess the effect of solver order, *i.e.*, m in Eq. (5), on *ConsistencySolver*'s preview consistency at 5, 8, and 10 steps. As shown in Tab. S2, Order 4 consistently achieves the best overall performance across step counts, leading in key structural and perceptual metrics while maintaining strong semantic alignment. Lower-order solvers (*e.g.*, Order 2 or 3) show reduced fidelity in layout and depth consistency, whereas Order 5 yields only marginal improvements in minor dimensions likely due to the increased RL search space complexity. Overall, Order 4 strikes a better balance between efficiency and complexity.

D.6. Ablation study on reward models

We investigate the impact of different reward models on the RL training of *ConsistencySolver*. As shown in Tab. S3, the Depth reward provides strong structural fidelity, consistently achieving good performance across all steps. Meanwhile, the Img. reward performs well in pixel-level fidelity, particularly at higher steps. Although CLIP and DINO show competitive results in semantic alignment, Depth offers a more balanced trade-off between structural consistency and overall robustness. We therefore adopt Depth as the default reward for its reliable generalization across diverse evaluation scenarios.

Table S3. Ablation study on reward model choice at 5, 8, and 10 steps. Best results per metric in **bold**.

Rewards	Steps	Dep.↑	Inc.↑	Seg.↑	Img.↑	CLIP↑	DINO↑
5 Steps							
Dep.	5	19.29	87.07	69.42	**20.75**	94.22	86.35
Inc.	5	19.20	87.05	69.49	20.18	94.29	86.30
CLIP	5	**19.32**	**87.30**	**69.73**	20.30	**94.46**	86.50
Img.	5	**19.32**	87.22	69.44	20.69	94.40	**86.53**
DINO	5	19.29	87.19	69.64	20.43	94.39	86.43
Seg.	5	19.16	86.81	69.28	19.85	94.12	86.01
8 Steps							
Dep.	8	**22.15**	91.65	**78.52**	**23.43**	**96.35**	**91.09**
Inc.	8	22.00	91.51	77.33	22.67	96.17	90.61
CLIP	8	21.94	91.45	77.54	22.56	96.15	90.75
Img.	8	22.11	**91.75**	78.17	23.39	96.34	90.97
DINO	8	22.03	91.62	77.84	22.99	96.28	90.87
Seg.	8	21.82	91.36	77.05	22.41	96.05	90.39
10 Steps							
Dep.	10	**24.94**	**93.88**	**83.22**	25.32	97.25	**93.48**
Inc.	10	24.17	93.35	82.01	24.33	97.05	92.68
CLIP	10	24.25	93.44	81.84	24.14	96.99	92.76
Img.	10	24.80	93.87	82.74	**25.37**	**97.28**	93.39
DINO	10	24.49	93.60	82.55	24.81	97.15	93.01
Seg.	10	23.73	93.15	81.37	24.04	96.96	92.44

Table S4. End-to-end inference time (seconds) across hardware configurations.

Hardware	GenEval		COCO 2017		LAION		Speedup
	Full	Preview	Full	Preview	Full	Preview	
Without user interaction time							
SD1.5 (H100)	3.82	2.16	3.52	2.03	5.18	2.58	1.84×
SD1.5 (M4)	135.68	69.99	126.13	66.81	197.60	83.78	2.06×
SD1.5 (CPU)	145.30	76.35	135.07	72.80	211.61	91.72	2.02×
With user interaction time (300 ms per trial)							
SD1.5 (H100)	4.89	3.30	4.51	3.06	6.73	4.22	1.52×
SD1.5 (M4)	136.75	71.13	127.12	67.84	199.15	85.42	2.04×
SD1.5 (CPU)	146.37	77.49	136.06	73.83	213.16	93.36	2.00×

E. Latency analysis across hardware configurations

We present a detailed comparison of preview-and-refine latency by incorporating user reaction time across different hardware configurations. Following prior studies on human visual perception [2], we assume a user reaction time of approximately 0.3 seconds per sample. Tab. S4 reports the end-to-end inference time on three datasets with different hardware. On high-performance GPUs (*e.g.*, H100), the overall speedup is reduced when accounting for user interaction time. However, when deploying on M4 chips and CPUs (Platinum 8480), the sampling time per trial is substantially longer, making the human evaluation overhead negligible in comparison. These results highlight that the efficiency gains of *Diffusion Preview* are particularly pronounced in resource-constrained deployment scenarios, where the preview-and-refine paradigm achieves up to 2.04× speedup even after accounting for user interaction time.

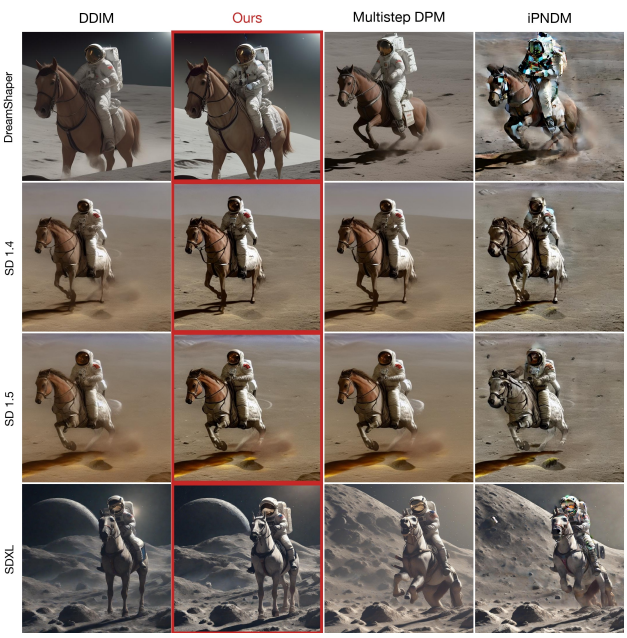

Figure S2. *ConsistencySolver* trained on SD1.5 generalizes effectively to SD1.4, DreamShaper, and SDXL, achieving superior visual quality compared to training-free ODE solvers.

F. Cross-model generalization

We provide qualitative comparisons for the cross-model generalization experiments discussed in the main text (Tab. 4). Fig. S2 shows that *ConsistencySolver* trained on SD1.5 produces visually superior results on SD1.4, DreamShaper, and SDXL compared to training-free ODE solvers, further confirming its strong transferability across model families.