ICCV HONOLULU HAWAII
OCT 19-23, 2025

Distilling Parallel Gradients for Fast ODE Solvers of Diffusion Models

Beier Zhu [1] Ruoyu Wang [2] Tong Zhao [2] Hanwang Zhang [1] Chi Zhang [2]

[1]Nanyang Technological University [2]Westlake University

Paper Code

1. Background

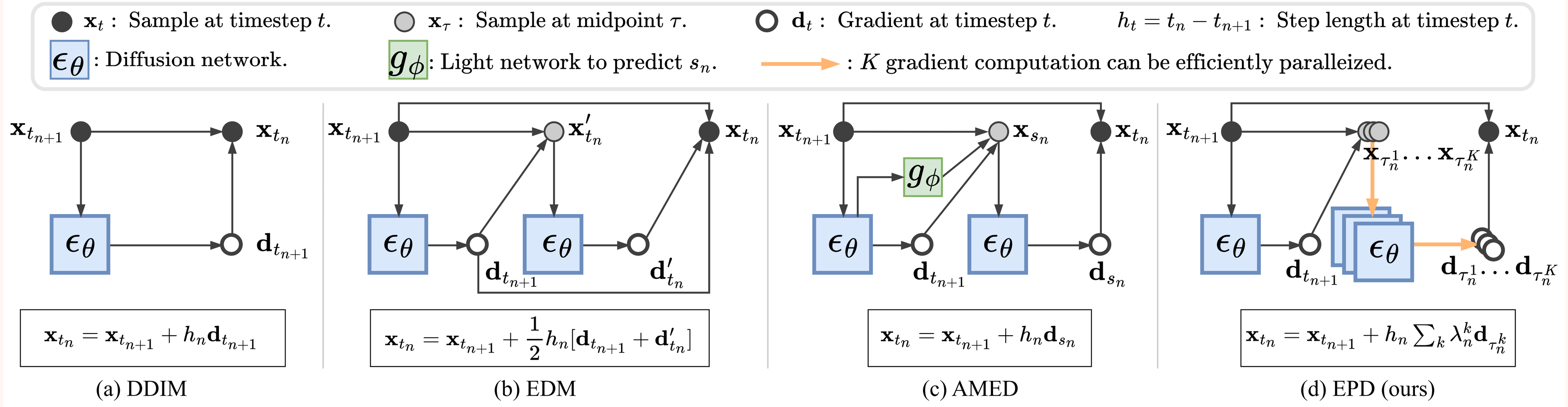
Diffusion models (DMs) have become a leading paradigm in generative modeling. DMs operate by gradually refining a noisy input through a denoising process, producing high-fidelity outputs. However, the multi-step sequential denoising process **introduces substantial latency**. Recent efforts on accelerating DMs typically fall into three categories.

- **Solver-based methods** develop fast numerical solvers to reduce sampling steps. However, inherent truncation errors lead to significant quality degradation when the number of function evaluations (NFE) is low (e.g., < 5).
- **Distillation-based methods** distill student DMs to generate high-quality samples within a minimal number of NFEs, often as low as one. However, this requires extensive training with carefully designed objectives, making the distillation process computationally expensive.
- **Parallelism-based methods** accelerate diffusion models by trading computation for speed. While promising, this direction under low NFEs remains underexplored.

2. Motivation

At a high level, various existing ODE solvers utilize gradients at different timesteps to approximate the ODE solution with varying accuracy. As shown in Figure below:

- DDIM [5] adopts the rectangle rule that uses the gradient at the start point $\mathbf{d}_{t_{n+1}}$.
- EDM [1] adopts the trapezoidal rule that averages the gradients of the start point $\mathbf{d}_{t_{n+1}}$ and the end point $\mathbf{d}'_{t_n}$.
- AMED [6] optimizes a small network g_ϕ to output an intermediate timestep $s_n \in (t_n, t_{n+1})$ to compute the gradient $\mathbf{d}_{s_n}$.



Compared to DDIM, EDM and AMED introduce an **additional timestep** for gradient computation (t_n and s_n), leading to improved integral estimation. The key motivation behind our EPD-Solver is to further leverage **multiple timesteps**:

- Compute the gradients at K intermediate timesteps **in parallel** $\mathbf{d}_{\tau_n^1}, \ldots, \mathbf{d}_{\tau_n^K}, \tau_n^k \in (t_{n+1}, t_n)$.
- Combine the K gradients via **convex aggregation**, yielding a more precise integral approximation $(\sum_k \lambda_n^k \mathbf{d}_{\tau_n^k})$
- As the computation of gradients are independent, **the latency remains unchanged**.

3. Theoretical Justification

The use of gradients estimated at multiple timesteps for improved integral approximation can be theoretically justified by the following mean value theorem for vector-valued functions [4]:

When f has values in an n-dimensional vector space and is continuous on the closed interval $[a, b]$ and differentiable on the open interval (a, b), we have

$$f(b) - f(a) = (b-a) \sum_{k=1}^{n} \lambda_k f'(c_k), \tag{1}$$

for some $c_k \in (a, b), \lambda_k \geq 0$, and $\sum_{k=1}^{n} \lambda_k = 1$.

In the context of denoising process, the function outputs an d-dimensional vector as $\mathbf{x} \in \mathbb{R}^d$. The exact integral of $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ over the interval $[t_n, t_{n+1}]$ can be expressed as a simplex-weighted combination of gradients evaluated at d intermediate points, scaled by the interval length $h_n = t_n - t_{n+1}$, as formulated in our EPD-Solver.

4. Method: EPD-Solver

Definition of parameters and inference

We define the parameters at step n as $\Theta_n = \{\tau_n^k, \lambda_n^k, \delta_n^k, o_n\}_{k=1}^K$:

- $\tau_n^k \in (t_{n+1}, t_n)$: k-th intermediate timestep.
- $\lambda_n^k \geq 0, \sum_k \lambda_n^k = 1$: combining weights.
- δ_n^k: k-th timestep perturbation.
- o_n: scaling perturbation. δ_n^k and o_n are proposed to mitigate exposure bias [2].

Update rule:

$$\mathbf{x}_{t_n} = \mathbf{x}_{t_{n+1}} + (1 + o_n) h_n \sum_{k=1}^K \lambda_n^k \boldsymbol{\epsilon}_\theta(\mathbf{x}_{\tau_n^k}, \tau_n^k + \delta_n^k)$$

Distillation-based optimization

- **Generating accurate teacher trajectory:** Given a student time schedule with N steps $\mathcal{T}_{\text{stu}} = \{t_0 = t_{\min}, ..., t_N = t_{\max}\}$, we insert M intermediate steps between t_n and t_{n+1}, i.e., $\mathcal{T}_{\text{tea}} = \{t_0, ..., t_n, t_n^1, ..., t_n^M, t_{n+1}, .., t_N\}$, The teacher trajectories are generated by any ODE solver (e.g, DPM-Solver) and store the reference states as $\{\mathbf{y}_{t_n}\}_{n=0}^N$.
- **Generate student trajectory:** We sample student trajectory $\{\mathbf{x}_{t_n}\}_{n=0}^N$ with the same initial noise $\mathbf{y}_{t_N}$, and use the update rule with the parameters $\{\Theta_n\}_{n=1}^N$.
- **Alignment:** We align the two trajectories w.r.t some distance measurement $\text{dist}(\cdot, \cdot)$: $\mathcal{L}_n(\Theta_{1:n}) = \text{dist}(\mathbf{x}_{t_n}, \mathbf{y}_{t_n})$.

EPD-Plugin to existing solvers

EPD-Solver augments existing solvers by substituting their gradient estimation with parallel branches. We illustrate this using the multi-step iPNDM [3]; see the main paper for details.

5. Experiments

	Method	(Para.) NFE 3	5	7	9
Single-step	DDIM	93.36	49.66	27.93	18.43
	EDM	306.2	97.67	37.28	15.76
	DPM-Solver-2	155.7	57.30	10.20	4.98
	AMED-Solver	18.49	7.59	4.36	3.67
Multi-step	DPM-Solver++(3M)	110.0	24.97	6.74	3.42
	UniPC	109.6	23.98	5.83	3.21
	iPNDM	47.98	13.59	5.08	3.17
	AMED-Plugin	10.81	6.61	3.65	2.63
Parallel	ParaDiGMS	51.03	18.96	7.18	6.19
	EPD-Solver (ours)	**10.40**	**4.33**	**2.82**	2.49
	EPD-Plugin (ours)	10.54	4.47	3.27	**2.42**

(a) Unconditional **CIFAR10** 32×32

	Method	(Para.) NFE 3	5	7	9
Single-step	DDIM	78.21	43.93	28.86	21.01
	EDM	356.5	116.7	54.51	28.86
	DPM-Solver-2	266.0	87.10	22.59	9.26
	AMED-Solver	47.31	14.80	8.82	6.31
Multi-step	DPM-Solver++(3M)	86.45	22.51	8.44	4.77
	UniPC	86.43	21.40	7.44	4.47
	iPNDM	45.98	17.17	7.79	4.58
	AMED-Plugin	26.87	12.49	6.64	4.24
Parallel	ParaDiGMS	43.64	20.92	16.39	8.81
	EPD-Solver (ours)	21.74	**7.84**	**4.81**	3.82
	EPD-Plugin (ours)	**19.02**	7.97	5.09	**3.53**

(b) Unconditional **FFHQ** 64×64

	Method	(Para.) NFE 3	5	7	9
Single-step	DDIM	82.96	43.81	27.46	19.27
	EDM	249.4	89.63	37.65	16.76
	DPM-Solver-2	140.2	42.41	12.03	6.64
	AMED-Solver	38.10	10.74	6.66	5.44
Multi-step	DPM-Solver++(3M)	91.52	25.49	10.14	6.48
	UniPC	91.38	24.36	9.57	6.34
	iPNDM	58.53	18.99	9.17	5.91
	AMED-Plugin	28.06	13.83	7.81	5.60
Parallel	ParaDiGMS	41.11	17.27	13.67	6.38
	EPD-Solver (ours)	**18.28**	**6.35**	5.26	4.27
	EPD-Plugin (ours)	19.89	8.17	**4.81**	**4.02**

(c) Conditional **ImageNet** 64×64

	Method	(Para.) NFE 3	5	7	9
Single-step	DDIM	86.13	34.34	19.50	13.26
	EDM	291.5	175.7	78.67	35.67
	DPM-Solver-2	210.6	80.60	23.25	9.61
	AMED-Solver	58.21	13.20	7.10	5.65
Multi-step	DPM-Solver++(3M)	111.9	23.15	8.87	6.45
	UniPC	112.3	23.34	8.73	6.61
	iPNDM	80.99	26.65	13.80	8.38
	AMED-Plugin	101.5	25.68	8.63	7.82
Parallel	ParaDiGMS	100.3	31.68	15.85	8.56
	EPD-Solver (ours)	**13.21**	**7.52**	5.97	5.01
	EPD-Plugin (ours)	14.12	8.26	**5.24**	**4.51**

(d) Unconditional **LSUN Bedroom** 256×256

See the main paper for qualitative results.

References

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

[2] Mingxiao Li, Tingyu Qu, Ruicong Yao, Wei Sun, and Marie-Francine Moens. Alleviating exposure bias in diffusion models through sampling with shifted time steps. In *ICLR*, 2024.

[3] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *ICLR*, 2022.

[4] Robert M McLeod. Mean value theorems for vector valued functions. *Proceedings of the Edinburgh Mathematical Society*, 14(3):197–209, 1965.

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[6] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *CVPR*, 2024.