A. Additional Implementation Details

A.1. Implementation Details of `EPD-Solver`

At each sampling step n (from t_{n+1} to t_n) in an N-step process, the solver provides a set of learned parameters $\Theta_n = \{\tau_n^k, \lambda_n^k, \delta_n^k, o_n\}_{k=1}^K$, implemented as follows:

Intermediate timesteps (τ_n^k): These are points within $[t_n, t_{n+1}]$, computed via geometric interpolation. Specifically, the interpolation ratio $r_n^k \in [0,1]$ is obtained by applying a sigmoid to a learnable scalar parameter, yielding

$$\tau_n^k = t_{n+1}^{r_n^k} \cdot t_n^{1-r_n^k}. \tag{13}$$

Simplex weights (λ_n^k): These non-negative weights form a convex combination of the K parallel gradients, satisfying $\sum_{k=1}^K \lambda_n^k = 1$. They are obtained by applying a softmax over K learnable scalar parameters.

Output scaling (o_n): A learnable scalar that scales the overall update direction by a factor of $(1 + o_n)$ to mitigate exposure bias between training and sampling. To implement this, we introduce a per-branch modulation term $\sigma_n^k \in [-0.05, 0.05]$ that scales the corresponding weight λ_n^k. Specifically, we constrain σ_n^k using a sigmoid-based transformation:

$$\sigma_n^k = 0.1 \times (\texttt{sigmoid}(\tilde{\sigma}_n^k) - 0.5),$$

where $\tilde{\sigma}_n^k$ is an unconstrained learnable parameter. The final scaling factor is then given by

$$o_n = \sum_k \lambda_n^k \sigma_n^k - 1.$$

Timestep shifting (δ_n^k): A trainable perturbation applied to the intermediate timestep τ_n^k, producing $\tau_n^k + \delta_n^k$ as input to the denoising network. We implement this by introducing a scaling factor s_n^k that transforms τ_n^k into $s_n^k \tau_n^k$. The relationship between s_n^k and δ_n^k is given by

$$s_n^k \tau_n^k = \tau_n^k + \delta_n^k \quad \Rightarrow \quad \delta_n^k = (s_n^k - 1)\tau_n^k.$$

To prevent overfitting, s_n^k is constrained to a small range (*e.g.*, $[0.95, 1.05]$) using a sigmoid-based transformation. Specifically, we map an unnormalized parameter $\tilde{s}_n^k$ as follows:

$$s_n^k = 1 + 0.1 \times (\texttt{sigmoid}(\tilde{s}_n^k) - 0.5).$$

A.2. Implementation Details of `EPD-Plugin`

The `EPD-Plugin` serves as a module integrated in any existing ODE solver. We illustrate this using the multi-step iPNDM [21, 50] sampler as a representative implementation. We begin with a brief review of the iPNDM sampler.

Review of iPNDM. Let $\mathbf{d}_t$ denote the estimated gradient at time step t, *i.e.*, $\mathbf{d}_t = \epsilon_\theta(\mathbf{x}_t, t)$. The update at time step t_n is given by:

Timesteps	Para. NFE			
	3	5	7	9
t_n, t_{n+1} (EDM)	306.2	97.67	37.28	15.76
$\sqrt{t_n t_{n+1}}, t_{n+1}$	129.6	16.51	9.86	7.06
$\frac{1}{2}(t_n + t_{n+1}), t_{n+1}$	105.8	36.14	18.08	9.85
$t_n, \sqrt{t_n t_{n+1}}$	225.5	130.8	78.49	44.38
$t_n, \frac{1}{2}(t_n + t_{n+1})$	198.6	119.6	59.23	32.21
$\sqrt{t_n t_{n+1}}, \frac{1}{2}(t_n + t_{n+1})$	136.1	21.17	10.80	5.83
random, t_{n+1}	90.8	30.01	14.37	9.14
random, random	110.7	57.1	22.86	11.91
`EPD-Solver`, $K = 2$	10.60	5.26	3.29	2.52

Table 7. FID results on the choices of two intermediate points. Evaluations are conducted on CIFAR-10 [15]. Start point: t_{n+1}, end point: t_n, midpoints: $\sqrt{t_n t_{n+1}}$, $\frac{1}{2}(t_n + t_{n+1})$, and 'random' denotes a midpoint randomly chosen from $[t_n, t_{n+1}]$.

$$\begin{aligned} \mathbf{d}'_{t_{n+1}} &= \tfrac{1}{24}(55\mathbf{d}_{t_{n+1}} - 59\mathbf{d}_{t_{n+2}} + 37\mathbf{d}_{t_{n+3}} - 9\mathbf{d}_{t_{n+4}}) \\ \mathbf{x}_{t_n} &= \mathbf{x}_{t_{n+1}} + h_n \mathbf{d}'_{t_{n+1}}. \end{aligned} \tag{14}$$

This rule applies for $n < N - 3$; for brevity, we present only this case. Other cases can be found in the original paper.

Our EPD plugin for iPNDM. Our plugin replaces $\mathbf{d}_{t_{n+1}}$ with a weighted combination of K parallel intermediate gradients to reduce truncation error. Similar to `EPD-Solver`, we introduce the parameters at step n as $\Theta_n = \{\tau_n^k, \lambda_n^k, \delta_n^k, o_n\}_{k=1}^K$. The gradient is now estimated as

$$\mathbf{d}^{\mathsf{EPD}}_{t_{n+1}} = (1 + o_n) \sum_{k=1}^K \lambda_n^k \boldsymbol{\epsilon}_\theta(\mathbf{x}_{\tau_n^k}, \tau_n^k + \delta_n^k). \tag{15}$$

Accordingly, the update for `EPD-Plugin` becomes:

$$\begin{aligned} \mathbf{d}'_{t_{n+1}} &= \tfrac{1}{24}(55\mathbf{d}^{\mathsf{EPD}}_{t_{n+1}} - 59\mathbf{d}_{t_{n+2}} + 37\mathbf{d}_{t_{n+3}} - 9\mathbf{d}_{t_{n+4}}) \\ \mathbf{x}_{t_n} &= \mathbf{x}_{t_{n+1}} + h_n \mathbf{d}'_{t_{n+1}}. \end{aligned} \tag{16}$$

`EPD-Plugin` incurs minimal training overhead, in line with the lightweight design of the `EPD-Solver`. Thanks to its limited number of learnable parameters, the optimization process is highly efficient.

A.3. Implementation Details of ParaDiGMS

For direct comparison with `EDP-{Solver, Plugin}`, we re-implemented the ParaDiGMS sampler [38] in the EDM [10] framework, as its public implementation[1] is tailored for Stable Diffusion. To ensure a fair latency comparison with our single-GPU `EPD-Solver`, we run ParaDiGMS on two NVIDIA 4090 GPUs, distributing the workload evenly by matching the Para. NFE/GPU ratio.

Specifically, to align the parallel structure with `EPD-Solver` ($K = 2$), we set the batch window size

[1]https://github.com/AndyShih12/paradigms

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	10.40	0	0	0.01339	0.96349	0.99731	0.85185
			1	0.67921	0.95231	0.99754	0.14815
		1	0	0.10020	1.03590	0.99500	0.75008
			1	0.28855	0.95457	1.02139	0.24992
5	4.33	0	0	0.03333	0.95415	0.99735	0.86941
			1	0.79558	0.95376	0.98616	0.13059
		1	0	0.07587	1.04503	0.99400	0.41741
			1	0.63244	1.04331	1.00711	0.58259
		2	0	0.38699	0.95588	1.00299	0.22410
			1	0.09434	1.01795	0.99999	0.77590
7	2.82	0	0	0.02511	0.96016	0.99725	0.86908
			1	0.91820	0.95206	1.01268	0.13092
		1	0	0.27815	0.98792	0.98996	0.80595
			1	0.81671	0.99280	1.01571	0.19405
		2	0	0.34431	1.03617	0.99038	0.17049
			1	0.60552	1.03999	0.98517	0.82951
		3	0	0.09416	1.01655	1.00019	0.77621
			1	0.41999	0.96088	1.00966	0.22379
9	2.49	0	0	0.28390	0.96336	0.99459	0.74143
			1	0.08408	1.01058	0.99785	0.25857
		1	0	0.33981	0.97201	0.99713	0.31062
			1	0.47617	0.98810	1.00195	0.68938
		2	0	0.61703	1.03201	0.99898	0.79387
			1	0.12204	1.01552	0.98848	0.20613
		3	0	0.58062	1.02698	0.99284	0.90470
			1	0.31738	1.02504	0.98079	0.09530
		4	0	0.08719	0.98858	0.99555	0.77554
			1	0.44045	0.97831	1.02114	0.22446

(a) **CIFAR10** 32×32 [15]

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	21.74	0	0	0.00472	0.95251	0.99909	0.85527
			1	0.61291	0.95212	1.00128	0.14473
		1	0	0.14636	1.00077	0.99866	0.90603
			1	0.52375	1.03973	1.00627	0.09397
5	7.84	0	0	0.00761	0.95240	0.98863	0.85668
			1	0.68196	0.95138	1.02573	0.14332
		1	0	0.48364	1.04868	1.01419	0.98053
			1	0.19897	1.03808	1.02313	0.01947
		2	0	0.51289	1.01520	0.99043	0.12838
			1	0.12570	0.96696	0.99892	0.87162
7	4.81	0	0	0.00344	0.95175	0.99173	0.89005
			1	0.90422	0.95040	1.01825	0.10995
		1	0	0.61922	1.03974	0.99767	0.62252
			1	0.06710	1.03036	1.00397	0.37748
		2	0	0.36516	1.03981	1.01085	0.49539
			1	0.71102	1.03331	1.01083	0.50461
		3	0	0.51302	0.99448	1.02493	0.15205
			1	0.11444	0.96889	0.99995	0.84795
9	3.82	0	0	0.07802	0.95010	0.99990	0.16419
			1	0.08710	0.95008	0.99990	0.83581
		1	0	0.85788	0.99068	0.98106	0.00087
			1	0.51685	0.99149	0.99980	0.99913
		2	0	0.5361	1.01276	0.99527	0.68458
			1	0.49629	1.01888	0.99385	0.31542
		3	0	0.55543	1.00901	1.00370	0.83477
			1	0.95208	1.01405	1.00179	0.16523
		4	0	0.10233	0.95959	0.99459	0.85282
			1	0.53488	1.03980	1.04863	0.14718

(b) **FFHQ** 64×64 [9]

Table 8. Optimized Parameters for `EPD-Solver` ($K = 2$) on CIFAR10 and FFHQ.

of ParaDiGMS to 2. The core principle was to adjust the tolerance parameter, ranging from 1×10^{-2} to 1×10^{-1}, to calibrate the total Para. NFE. The ratio of Para. NFE / GPUs was set to 3, 5, 7 and 9, which ensures the per-GPU workload and latency level for ParaDiGMS roughly matches the single-GPU `EPD-Solver`. We also observed that the efficiency of ParaDiGMS is reduced in low-NFE regimes, as the substantial error per iteration causes its solver stride to frequently set to 1.

B. Additional Experimental Results

Other choice of intermediate points. In Tab. 7, we compare our `EPD-Solver` with $K = 2$, *i.e.*, two learned intermediate points, against two manually selected midpoints and randomly selected ones. In particular, the manually selected midpoints include the start timestep t_n, the end timestep t_{n+1} (adopted in EDM), the geometric mean $\sqrt{t_n t_{n+1}}$ (used in DPM-Solver-2), and the arithmetic mean $\frac{1}{2}(t_n + t_{n+1})$. The random midpoints are uniformly sampled from $[t_n, t_{n+1}]$. We note several observations: (1) The combination of start points with mean points (geometric and arithmetic) significantly outperforms combinations that include the end point. For example, using the geometric and arithmetic points achieves an FID of 5.83 with NFE = 9, whereas incorporating the end point leads to much higher FID scores — 44.38 and 32.21 for the geometric and arithmetic points, respectively. (2) The combination that includes random points achieves competitive results. For instance, using a random point together with the start point yields better FID scores than EDM across all NFE values. (3) The gap between the best combination of handcrafted intermediate timesteps and our learned ones remains large, highlighting the necessity of our proposed method.

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	18.28	0	0	0.03892	0.90820	0.99810	0.78701
			1	0.58080	0.95077	1.00097	0.21299
		1	0	0.18326	0.99336	0.99910	0.97757
			1	0.08246	1.01142	1.02640	0.02243
5	6.35	0	0	0.14336	0.90835	0.99266	0.78550
			1	0.54204	0.93916	0.99114	0.21450
		1	0	0.71830	1.08078	1.00955	0.49788
			1	0.39094	1.07179	1.01071	0.50212
		2	0	0.25820	0.96964	1.00597	0.37857
			1	0.10124	1.00380	1.00316	0.62143
7	5.26	0	0	0.11952	0.90686	0.99347	0.91217
			1	0.95726	0.91100	1.01887	0.08783
		1	0	0.41813	1.03421	0.99877	0.83649
			1	0.76716	1.04605	1.00396	0.16351
		2	0	0.86120	1.03538	1.00931	0.02866
			1	0.52961	1.04485	1.00040	0.97134
		3	0	0.19129	0.98157	1.0024	0.99873
			1	0.17888	0.99072	1.02263	0.00127
9	4.27	0	0	0.97878	0.90410	1.01060	0.04239
			1	0.12206	0.90047	0.99891	0.95761
		1	0	0.40113	0.97924	0.99857	0.90324
			1	0.84037	1.04647	0.99850	0.09676
		2	0	0.55210	1.00744	0.99590	0.99983
			1	0.17699	0.97798	1.01484	0.00017
		3	0	0.67823	0.99619	1.01995	0.99919
			1	0.89296	1.02559	1.02289	0.00081
		4	0	0.26663	0.91395	1.01391	0.60252
			1	0.00584	1.06452	1.00333	0.39748

(a) **ImageNet** 64×64 [33]

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	13.21	0	0	0.82995	0.98769	1.01204	0.09938
			1	0.0410	1.0101	0.9989	0.9006
		1	0	0.03654	1.00350	0.98716	0.01419
			1	0.22279	0.97061	1.00927	0.98581
5	7.52	0	0	0.99712	1.00000	0.99752	0.07831
			1	0.02895	1.00000	1.00046	0.92169
		1	0	0.52144	1.00000	1.00186	0.61657
			1	0.18287	1.00000	0.99460	0.38343
		2	0	0.20350	1.00000	0.96961	0.24707
			1	0.23099	1.00000	1.00159	0.75293
7	5.97	0	0	0.92247	1.00000	1.00783	0.00004
			1	0.02283	1.00000	0.99966	1.00000
		1	0	0.45881	1.00000	1.00193	0.46663
			1	0.54699	1.00000	1.00185	0.53337
		2	0	0.09864	1.00000	0.98422	0.06541
			1	0.46885	1.00000	0.99675	0.93459
		3	0	0.20864	1.00000	0.96134	0.98301
			1	0.09425	1.00000	1.02840	0.01699
9	5.01	0	0	0.87854	1.00000	1.00569	0.07317
			1	0.07964	1.00000	0.99953	0.92683
		1	0	0.40848	1.00000	0.99842	0.82916
			1	0.94301	1.00000	1.00355	0.17084
		2	0	0.67654	1.00000	1.00375	0.01636
			1	0.49911	1.00000	1.00348	0.98364
		3	0	0.45169	1.00000	0.98647	0.14504
			1	0.40655	1.00000	0.99226	0.85496
		4	0	0.30053	1.00000	1.00438	0.02853
			1	0.20058	1.00000	0.95733	0.97147

(b) **LSUN Bedroom** 256×256 [49]

Table 9. Optimized Parameters for `EPD-Solver` ($K = 2$) on ImageNet and LSUN Bedroom.

B.1. Optimized Parameters for `EPD-Solver`

We provide our optimized parameters of `EPD-Solver` with $K = 2$ for CIFAR-10, ImageNet, FFHQ and LSUN Bedroom in Tabs. 8 and 9 with different Para. NFEs. According to the implementation details in Suppl. A.1, the parameters $\tau_n^k, \delta_n^k, o_n$ are derived as follows:

$$\tau_n^k = t_{n+1}^{r_n^k} \cdot t_n^{1-r_n^k} \tag{17}$$

$$\delta_n^k = (s_n^k - 1)\tau_n^k \tag{18}$$

$$o_n = \sum_k \lambda_n^k \sigma_n^k - 1 \tag{19}$$

B.2. Optimized Parameters for `EPD-Plugin`

We provide our optimized parameters of `EPD-Plugin` with $K = 2$ for CIFAR10, ImageNet, FFHQ and LSUN Bedroom in Tabs. 10 and 11 with different Para.NFEs.

B.3. Additional Qualitative Results

Here, we show some qualitative results on different datasets in Figs. 7 to 10.

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	10.54	0	0	0.06837	0.81145	0.99957	0.91271
			1	0.68803	0.85836	0.99981	0.08729
		1	0	0.12320	0.97533	0.99903	0.85072
			1	0.28206	0.85043	1.00671	0.14928
5	4.47	0	0	0.10548	0.80808	0.99606	0.95656
			1	0.96750	0.89210	1.00082	0.04344
		1	0	0.04114	1.03816	1.00480	0.52907
			1	0.57891	1.02063	1.02490	0.47093
		2	0	0.27989	1.00150	0.95600	0.26331
			1	0.05394	1.02182	0.98523	0.73669
7	3.27	0	0	0.08991	0.80504	0.99845	0.94689
			1	0.94988	0.95487	1.01496	0.05311
		1	0	0.04569	0.88770	0.99774	0.75623
			1	0.80305	1.04391	0.99378	0.24377
		2	0	0.91959	1.10578	0.99989	0.00408
			1	0.42678	1.01745	1.00242	0.99592
		3	0	0.36480	0.90472	1.02327	0.20787
			1	0.07649	0.96814	1.00433	0.79213
9	2.42	0	0	0.08244	0.80210	0.99483	0.08638
			1	0.25440	0.81528	0.99964	0.91362
		1	0	0.02193	0.80719	0.99517	0.99163
			1	0.02935	0.88719	0.99437	0.00837
		2	0	0.25227	1.08671	0.99438	0.02010
			1	0.55490	1.03722	0.99923	0.97990
		3	0	0.48861	1.01472	1.00312	0.81266
			1	0.02553	0.98693	1.00521	0.18734
		4	0	0.07257	0.97384	0.99552	0.78925
			1	0.39513	0.96933	0.99003	0.21075

(a) **CIFAR10** 32×32 [15]

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	19.02	0	0	0.07642	0.84410	0.99934	0.94986
			1	0.91510	0.97713	1.01079	0.05014
		1	0	0.17864	0.97337	1.00023	0.99041
			1	0.15293	0.90787	1.02719	0.00959
5	7.97	0	0	0.00858	0.82007	0.99986	0.87461
			1	0.65658	0.86946	0.99954	0.12539
		1	0	0.39945	0.99765	1.00157	0.99812
			1	0.18867	1.03054	1.01357	0.00188
		2	0	0.33148	0.96555	0.99766	0.22642
			1	0.07594	0.97690	0.99730	0.77358
7	5.09	0	0	0.01069	0.81532	0.99965	0.92015
			1	0.85634	0.86078	0.99965	0.07985
		1	0	0.37517	1.00369	0.99838	0.88685
			1	0.71151	1.00119	1.00481	0.11315
		2	0	0.08475	1.04325	1.03287	0.00052
			1	0.38954	1.00524	1.00463	0.99948
		3	0	0.08461	0.98373	0.98399	0.76003
			1	0.39386	1.01515	0.97975	0.23997
9	3.53	0	0	0.94960	0.82963	1.00126	0.06572
			1	0.00362	0.82194	0.9998	0.93428
		1	0	0.06822	0.87369	0.99903	0.19003
			1	0.48656	1.01113	0.99772	0.80995
		2	0	0.38262	1.02269	0.99920	0.84123
			1	0.98681	0.99794	1.01047	0.15877
		3	0	0.08146	0.99005	1.01881	0.56715
			1	0.89689	1.01201	0.99138	0.43285
		4	0	0.07455	0.96557	0.97884	0.80133
			1	0.47558	1.09918	0.95222	0.19867

(b) **FFHQ** 64×64 [9]

Table 10. Optimized Parameters for `EPD-Plugin` ($K = 2$) on CIFAR10 and FFHQ.

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	19.89	0	0	0.01805	0.89265	0.99984	0.81070
			1	0.59732	0.95910	0.99862	0.18930
		1	0	0.15989	0.96659	1.00771	0.96197
			1	0.26658	0.89747	1.04079	0.03803
5	8.17	0	0	0.11246	0.82261	0.99876	0.92199
			1	0.92205	0.96191	1.01100	0.07801
		1	0	0.00511	0.97233	0.99878	0.45635
			1	0.61007	0.99912	1.00419	0.54365
		2	0	0.35416	0.92432	0.99057	0.04391
			1	0.13234	0.96354	0.99885	0.95609
7	4.81	0	0	0.14306	0.82532	0.99963	0.99640
			1	0.02764	0.94802	0.96580	0.00360
		1	0	0.46578	0.98602	1.00224	0.99615
			1	0.09086	1.08617	1.02104	0.00385
		2	0	0.04504	1.05987	1.01408	0.00020
			1	0.44154	0.99292	0.99536	0.99980
		3	0	0.03175	0.90298	0.98815	0.00276
			1	0.14969	0.94543	1.00853	0.99724
9	4.02	0	0	0.33263	0.84332	0.99983	0.12259
			1	0.13371	0.85792	0.99931	0.87741
		1	0	0.05410	0.89662	1.00055	0.24089
			1	0.54876	0.99484	0.99886	0.75911
		2	0	0.37444	1.00578	1.00105	0.88450
			1	0.94384	1.01652	0.98910	0.11550
		3	0	0.28771	1.00243	0.99434	0.76097
			1	0.82883	1.00291	0.99311	0.23903
		4	0	0.11117	0.98196	1.01350	0.80293
			1	0.41243	0.88880	1.08111	0.19707

(a) **ImageNet** 64×64 [33]

Para. NFE	FID	n	k	r_n^k	s_n^k	σ_n^k	λ_n^k
3	14.12	0	0	0.78697	1.00000	1.00375	0.10230
			1	0.02085	1.00000	0.99945	0.89770
		1	0	0.08334	1.00000	0.96782	0.18352
			1	0.23899	1.00000	0.99524	0.81648
5	8.26	0	0	0.97220	0.98923	1.00016	0.07808
			1	0.03306	1.00415	0.99991	0.92192
		1	0	0.52337	0.99607	1.00463	0.60203
			1	0.01602	1.00079	0.99249	0.39797
		2	0	0.12524	0.99813	0.96174	0.49642
			1	0.29699	0.99950	1.01130	0.50358
7	5.24	0	0	0.97094	0.98527	1.01234	0.06101
			1	0.07156	1.00461	0.99893	0.93899
		1	0	0.70513	0.99016	1.01166	0.32484
			1	0.24738	0.98946	0.99696	0.67516
		2	0	0.27565	1.01344	0.97876	0.57267
			1	0.54473	1.00123	1.00931	0.42733
		3	0	0.16616	0.98549	0.96569	0.85584
			1	0.38606	0.99734	1.02813	0.14416
9	4.51	0	0	0.17020	1.01750	0.99792	0.34563
			1	0.01271	0.99479	1.00060	0.65437
		1	0	0.43953	0.98534	0.99969	0.96036
			1	0.82230	0.99246	0.99977	0.03964
		2	0	0.25682	1.00056	1.00433	0.30549
			1	0.50732	1.00773	0.99838	0.69451
		3	0	0.29627	1.01221	0.98564	0.31065
			1	0.48616	1.01091	0.99254	0.68935
		4	0	0.32949	1.00615	0.98884	0.04682
			1	0.19802	0.98760	0.95685	0.95318

(b) **LSUN Bedroom** 256×256 [49]

Table 11. Optimized Parameters for `EPD-Plugin` ($K = 2$) on ImageNet and LSUN Bedroom.

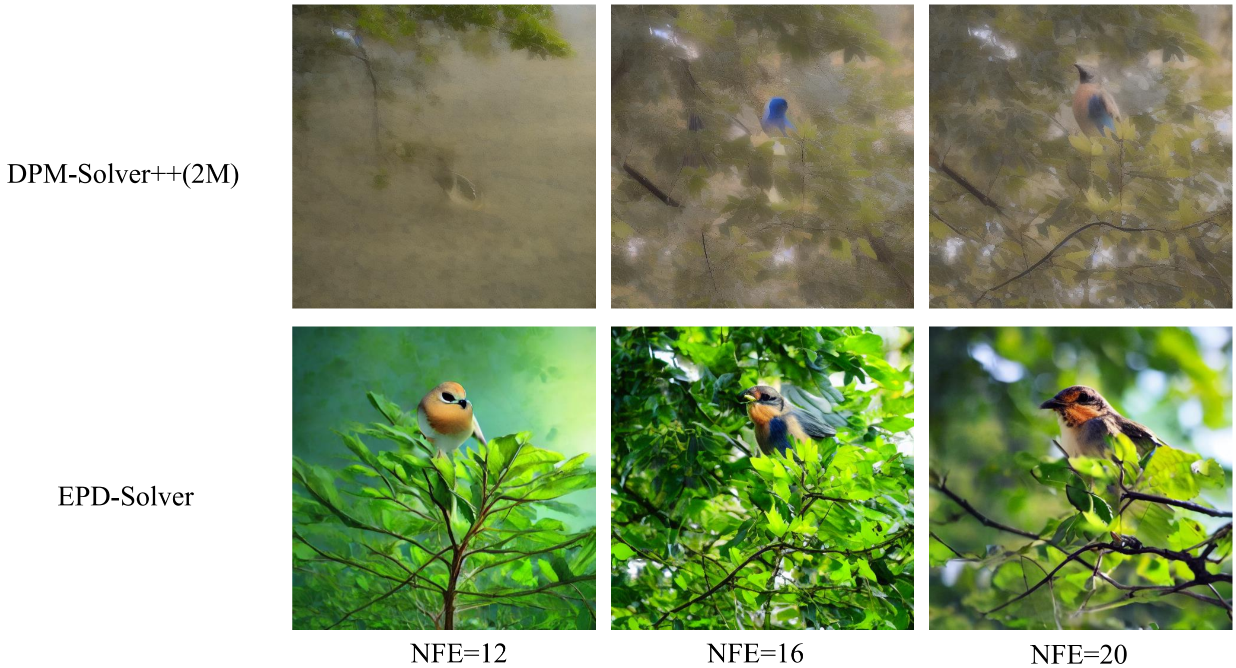


Figure 7. Comparison of image generation quality between DPM-Solver++ (2M) and `EPD-Solver`at different (Para.) NFEs.

(a) DPM-Solver-2. NFE=3

(b) DPM-Solver-2. NFE=9

(c) `EPD-Solver`. Para. NFE=3

(d) `EPD-Solver`. Para. NFE=9

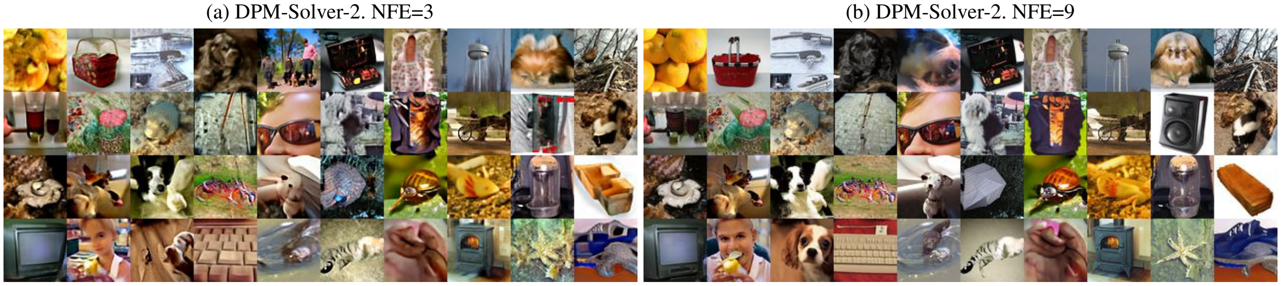
Figure 8. Qualitative result on CIFAR10 32×32 (3 and 9 NFEs)

(a) DPM-Solver-2. NFE=3 (b) DPM-Solver-2. NFE=9

(c) `EPD-Solver`. Para. NFE=3 (d) `EPD-Solver`. Para. NFE=9

Figure 9. Qualitative result on FFHQ 64×64 (3 and 9 NFEs)

(a) DPM-Solver-2. NFE=3 (b) DPM-Solver-2. NFE=9



(c) `EPD-Solver`. Para. NFE=3 (d) `EPD-Solver`. Para. NFE=9

Figure 10. Qualitative result on ImageNet 64×64 (3 and 9 NFEs)