PEEK: One-Step-Look-Ahead Exposure Bias Correction for Diffusion Sampling

Yunwei Bai* **Ying Kiat Tan** **Tsuhan Chen**
National University of Singapore
baiyunwei@u.nus.edu, yingkiat@u.nus.edu, dprtchen@nus.edu.sg

Abstract

Diffusion image generators owe much of their practical success to accurate denoising networks and strong numerical samplers. However, during sampling, the denoiser predicts the next step from flawed inputs that differ substantially from real data. This train-test gap, or exposure bias, caps image generation quality (i.e., blurry images, missing details, or distorted shapes). To mitigate the problem, we propose PEEK, a training-free deterministic sampler plugin. At each eligible denoising step, PEEK first performs a provisional one-step update, evaluates the denoiser once at the look-ahead state, and interpolates the current clean prediction with this one-step-later prediction to form the next noisy state. By taking the additional step, we obtain a generally more image-like prediction from the higher-signal noisy state, mitigating exposure bias. To optimize the interpolation positions, PEEK searches for an interpolation schedule with a greedy metric-driven search made efficient by the PEEK prefix and baseline noise-schedule-rebasing caches. Across pixel-space and latent-space diffusion models with unconditional, label-guided, and text-guided settings on CIFAR-10, LSUN Bedroom, ImageNet, and MS-COCO, the look-ahead correction consistently improves over baselines matched by number of function evaluations (NFE), with Fréchet Inception Distance (FID) reductions up to 49.3%. The one-time per-model calibration, the only extra cost, takes at most 2.2 minutes in our experiments. Project website is available at https://dachengruoque.github.io/PEEK-Video/.

1 Introduction

Diffusion models have become a dominant paradigm for image generation, combining a supervised denoising objective with an iterative generation process that gradually transports noise to data [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song et al., 2021b]. Their practical progress has come from both sides of the system: more capable denoising models and representations [Dhariwal and Nichol, 2021, Rombach et al., 2022, Peebles and Xie, 2023], and increasingly accurate samplers that treat generation as the numerical solution of a reverse-time stochastic or ordinary differential equation [Song et al., 2021a, Karras et al., 2022, Lu et al., 2022a,b, Zhang and Chen, 2023, Zhao et al., 2023].

Despite this progress, diffusion sampling remains sensitive to errors made along the denoising trajectory, which is most visible in the low-number-of-function-evaluations (NFE) regime, where a small number of large steps must convert an initial high-noise state into a noise-free image. Training typically supervises the network on forward-corrupted clean data, while inference applies the same network to states produced by previous flawed model predictions. The resulting states do not follow the same conditional distributions encountered during training [Ning et al., 2023]. Figure 1 gives a visual instance of the PEEK trajectory: the denoiser's prediction tends to be low-quality especially at the high-noise steps, though the training-time counterparts are real high-quality images. The final

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

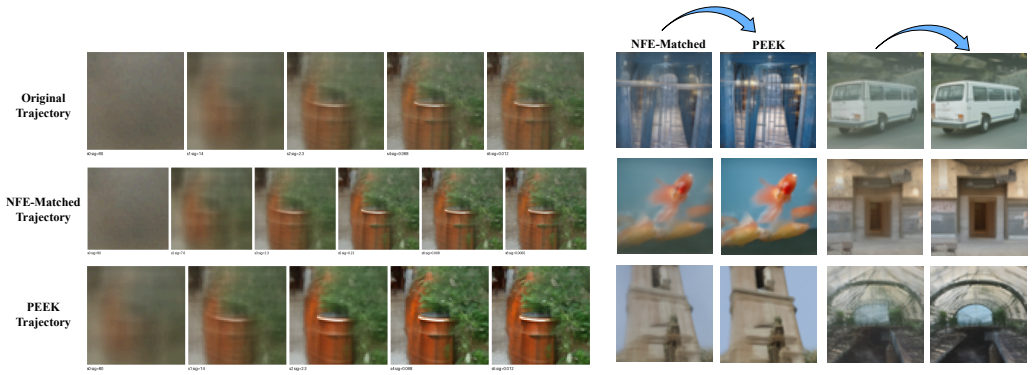

Figure 1: PEEK performs a provisional one-step update, and uses the look-ahead prediction to interpolate the current prediction at selected positions. The selected clean prediction is then used to construct the next noisy state. The trajectories illustrate that the early-step predictions tend to be especially low-quality, and PEEK can yield higher-quality images. Best viewed in color.

consequences of the exposure bias include limited generation quality, such as blurry images lacking details of those in the real images.

Existing inference-side methods mostly address the problem indirectly. Fast ODE solvers reduce discretization error by deriving higher-order updates [Lu et al., 2022a,b]. Exponential-integrator and predictor-corrector designs pursue the same low-NFE goal with problem-adapted updates [Zhang and Chen, 2023, Zhao et al., 2023, Karras et al., 2022]. Heun-type predictor-corrector samplers, for example, first form an Euler predictor and then mix the ODE derivatives estimated at the current and predicted states, usually by averaging the two slopes. Schedule optimization methods choose better time grids for a fixed sampler [Sabour et al., 2024]. Restart sampling adds stochastic noise and then re-denoises to contract accumulated sampling errors, and requires selecting restart intervals and repetition counts [Xu et al., 2023]. Thus prior training-free improvements primarily change the integration rule, change the timestep grid, or insert a stochastic re-denoising segment. PEEK targets, to our knowledge, a simple, **important** but **underexplored** correction mechanism: *what if we query the denoiser at a deterministic provisional next state and inject the resulting one-step-later clean prediction back to correct exposure bias, instead of just increasing the standard inference steps, for image generation quality improvement?*

In short, PEEK is a one-step-look-ahead correction for the deterministic sampling trajectory. Given a current noisy state, PEEK computes the usual clean prediction, takes a provisional next step, obtains the next clean prediction, and then uses an interpolation gate to form the new noisy state as input for the next-step model prediction. We additionally study a solver-direction variant, PEEK-G, for DPM-Solver-style samplers as an extension. Instead of interpolating the clean prediction used to form the next noisy state, PEEK-G interpolates the current solver direction with the direction implied by the one-step-later prediction. So far, extensive work has demonstrated that the generation quality improves along the denoising trajectory, which is expected as the later-step predictions are based on the higher signal-to-noise-ratio input. Furthermore, interchanging the current prediction with the one-step-look-ahead prediction does not break the inherent logic behind deterministic diffusion, as different time steps share the same real-image input and supervision target during training, where only the noise levels differ [Ho et al., 2020, Song et al., 2021a, Karras et al., 2022, Lu et al., 2022b, Zhang and Chen, 2023, Zhao et al., 2023]. Intuitively speaking, assuming perfect corrections that minimize the train-test gap, the otherwise-drifting denoising trajectory can reach the optimal track characterized by only the empirical training error. In contrast, if we only densify the denoising trajectory by increasing the number of ordinary solver steps, we may continue from the already-drifting trajectory where the wide early-inference train-test gaps persist throughout. Therefore, PEEK has the comparative advantage.

On the other hand, each active interpolation spends an extra denoiser evaluation and may exacerbate bias. Therefore, interpolation positions matter. We avoid manual hyperparameter tuning through a schedule search, with a greedy search selecting per-step interpolation gates using a target metric such

as FID [Heusel et al., 2017], and the NFE-matched guardrail rejecting look-ahead edits that are a weaker use of the same extra evaluation than an ordinary solver step. To make this search efficient, we exploit the cumulative structure of deterministic denoising, where 1) an NFE-matched baseline noise-schedule-rebasing cache and 2) a PEEK prefix cache avoid rerunning unchanged prefix noise states. Our contributions are:

- Against the exposure bias problem in low-NFE deterministic diffusion sampling, we propose PEEK, a training-free deterministic one-step-look-ahead interpolation that edits the current prediction to form the noisy state for the next-step inference.
- We extend the same look-ahead principle to PEEK-G, a DPM-Solver-compatible solver-direction variant.
- We introduce a metric-driven interpolation schedule search for PEEK(-G), together with baseline noise-schedule-rebasing and PEEK prefix caches that reduce the 9-step vanilla search count from 225 to 78 denoiser evaluations, a 65.3% reduction.
- We evaluate PEEK and PEEK-G on pixel-space and latent-space generation settings, including CIFAR-10 [Krizhevsky, 2009], LSUN Bedroom [Yu et al., 2015], ImageNet [Deng et al., 2009], and MS-COCO [Lin et al., 2014], and demonstrate nontrivial improvements over the NFE-matched baselines, with FID reductions up to 49.3%, under matched denoiser-evaluation budgets.

2 Related Work

Fast Diffusion Sampling. Diffusion image generation combines denoising models [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song et al., 2021b] with numerical samplers. Low-NFE generation has been improved through deterministic sampling paths, higher-order ODE solvers, exponential integrators, predictor-corrector updates, and schedule design [Song et al., 2021a, Liu et al., 2022, Lu et al., 2022a,b, Zhang and Chen, 2023, Zhao et al., 2023, Karras et al., 2022, Sabour et al., 2024]. These methods mainly change the integration rule or the timestep grid. Predictor-corrector samplers instantiate this idea at different levels: Heun's second-order sampler takes an Euler predictor, reevaluates the ODE slope at the predicted endpoint, and applies a trapezoidal average of the two slopes [Karras et al., 2022], while UniPC derives unified predictor and corrector formulas in half-log-SNR space, using current and previous model outputs so the corrector can raise solver order without an extra model evaluation [Zhao et al., 2023]. In contrast, PEEK interpolates the clean-sample prediction with the one-step-look-ahead counterpart.

Inference Mismatch and Correction. Other approaches reduce few-step error by distilling trajectories or adapting the model for short sampling [Salimans and Ho, 2022, Song et al., 2023, Oganov et al., 2025, Cai et al., 2026]. Training-free inference corrections are closer to our setting. Restart sampling inserts forward-noising and re-denoising segments to contract accumulated error [Xu et al., 2023], while SE2P uses information from adjacent denoising times in a parallel-processor design [Cisneros-Velarde, 2025]. PEEK instead spends selected extra evaluations on a deterministic one-step look-ahead query and compares against ordinary NFE-matched trajectories. The appendix gives more related work.

3 Method

3.1 Vanilla PEEK

We describe PEEK for deterministic first-order samplers, the setting used for the DPM-Solver++, DEIS, and UniPC wrapper experiments [Lu et al., 2022b, Zhang and Chen, 2023, Zhao et al., 2023]. Appendix gives preliminaries. Let $t_0 > t_1 > \cdots > t_{N-1}$ be decreasing timesteps and z_i the sampler state at t_i. We write the ordinary clean prediction as

$$\hat{x}_{0,i}^{\text{base}} = D_\theta(z_i, t_i). \tag{1}$$

A deterministic solver step is

$$z_{i+1} = \Phi_i(z_i, \hat{x}_{0,i}^{\text{base}}), \tag{2}$$

where Φ_i includes the sampler-specific conversion from prediction to the next noisy state.

One-Step Look-Ahead Interpolation. At step i, PEEK first computes the ordinary clean prediction $\hat{x}_{0,i}^{\text{base}}$ and an ordinary provisional next state

$$\tilde{z}_{i+1} = \Phi_i(z_i, \hat{x}_{0,i}^{\text{base}}). \tag{3}$$

It then evaluates the denoiser once more at the next solver timestep:

$$\hat{x}_{0,i}^{\text{look}} = D_\theta(\tilde{z}_{i+1}, t_{i+1}). \tag{4}$$

The clean prediction is passed through a binary gate:

$$\hat{x}_{0,i}^{\text{sel}} = (1-\gamma_i)\hat{x}_{0,i}^{\text{base}} + \gamma_i \hat{x}_{0,i}^{\text{look}}, \tag{5}$$

where $\gamma_i \in \{0, 1\}$. The final step starts from the original state z_i but uses the selected clean prediction:

$$z_{i+1}^{\text{PEEK}} = \Phi_i(z_i, \hat{x}_{0,i}^{\text{sel}}). \tag{6}$$

Let q_t be the forward-noised data marginal and p_i the sampler marginal at the same timestep. The denoiser is trained under q_{t_i} but queried under p_i. If $\ell_i(x) = \|D_\theta(x, t_i) - m_i(x)\|^2$ is the clean-prediction loss relative to $m_i(x) = \mathbb{E}_q[x_0 \mid x_{t_i} = x]$, the trajectory mismatch induces excess risk

$$\Delta_i = \mathbb{E}_{x\sim p_i}\ell_i(x) - \mathbb{E}_{x\sim q_{t_i}}\ell_i(x). \tag{7}$$

Large low-NFE steps can make Δ_i positive because p_i contains model-generated states away from the forward-noised data distribution. Let $x_0^\star$ be a reference clean target and L_i a local Lipschitz constant of the deterministic step with respect to its clean prediction. Then $\|\Phi_i(z, \hat{x}_0) - \Phi_i(z, x_0^\star)\| \le L_i\|\hat{x}_0 - x_0^\star\|$, so reducing clean-prediction error tightens the corresponding upper bound on one-step state error.

Diffusion training ties predictions at different noise levels to the same clean target [Ho et al., 2020, Song et al., 2021b,a]. The look-ahead query uses a higher-SNR state while the update remains anchored at z_i, so the correction is local and can propagate through later deterministic steps.

3.2 Interpolation Schedule Search

A fully interpolated PEEK can spend unnecessary denoiser calls. We therefore learn a schedule $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{N-2})$ for the $N-1$ peekable gates; the final step has no later noisy timestep.

With candidate set $\mathcal{A} = \{0, 1\}$, the search greedily chooses one gate at a time. Let $M(\boldsymbol{\gamma})$ be the target validation metric, FID in our experiments. At step i, previous gates are fixed and future gates are zero:

$$M_i(\gamma) = M(\gamma_0, \ldots, \gamma_{i-1}, \gamma, 0, \ldots, 0).$$

Let M_i^{base} be the metric of the ordinary solver at the denoiser-evaluation budget obtained by activating the proposed gate, including all previously selected nonzero gates. We evaluate each $\gamma \in \mathcal{A}$ and activate the gate only when it passes both comparisons:

$$\gamma_i = \begin{cases} 1, & \text{if } M_i(1) < \min\{M_i(0), M_i^{\text{base}}\}, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The guardrail therefore compares an active correction with spending its extra evaluation on an ordinary solver step. The zero candidate retains the current partial schedule at its lower budget.

For each candidate position, the search reuses the selected prefix state, evaluates the NFE-matched baseline, evaluates the current partial schedule, and tests each candidate gate. The selected state becomes the next prefix. Appendix gives pseudocode and a real search-log excerpt.

A nonzero γ_i means the active candidate outperforms both the current partial schedule and the NFE-matched ordinary baseline. A zero means the active candidate fails at least one of these two comparisons.

The search is a one-time calibration for a model, sampler, step count, and target metric. At deployment, the schedule is fixed. A PEEK sample with N solver steps and K nonzero gates uses $N + K$ denoiser evaluations, and our main comparisons use ordinary baselines with the same budget.

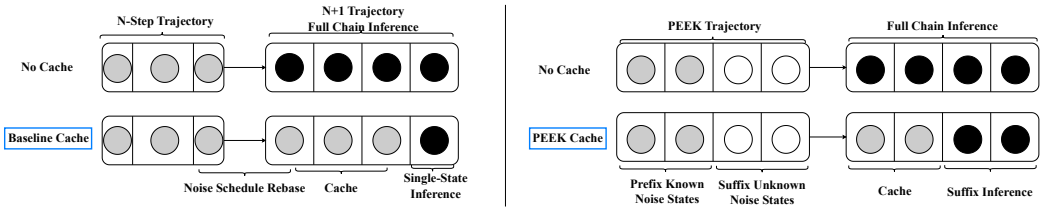

Figure 2: Caches used during schedule search. Baseline rebasing avoids rerunning ordinary prefixes as the NFE-matched chain grows, while the PEEK prefix cache lets later candidates run only the remaining suffix. Best viewed in color.

3.3 PEEK-G Extension

PEEK-G forms the same provisional one-step-look-ahead state as PEEK, and blends the current and look-ahead solver directions. The appendix gives the formal DPM-Solver direction notation.

3.4 Search-Time Cache Optimization

PEEK uses two caches for deterministic cumulative first-order search. Figure 2 summarizes their complementary reuse paths.

Baseline Noise-Schedule Rebasing Cache. For a deterministic sampler state z_t and clean prediction $\hat{x}_0$, the variance-preserving implied noise is

$$\hat{\epsilon} = \frac{z_t - \sqrt{\bar{\alpha}_t}\hat{x}_0}{\sqrt{1-\bar{\alpha}_t}}. \tag{9}$$

For target timestep s, the rebased state is

$$z_s = \sqrt{\bar{\alpha}_s}\hat{x}_0 + \sqrt{1-\bar{\alpha}_s}\hat{\epsilon}. \tag{10}$$

Equation (10) moves cached ordinary prefixes to the next NFE-matched grid without recomputing them. Substituting Eq. (9) into Eq. (10) gives

$$\begin{aligned} R_{t\to s}(z_t, \hat{x}_0) &= \sqrt{\bar{\alpha}_s}\hat{x}_0 + \sqrt{\frac{1-\bar{\alpha}_s}{1-\bar{\alpha}_t}}\left(z_t - \sqrt{\bar{\alpha}_t}\hat{x}_0\right) \\ &= c_x z_t + c_0 \hat{x}_0, \end{aligned} \tag{11}$$

where $c_x = \sqrt{(1-\bar{\alpha}_s)/(1-\bar{\alpha}_t)}$ and $c_0 = \sqrt{\bar{\alpha}_s} - c_x\sqrt{\bar{\alpha}_t}$. Thus a cached pair $(z_t, \hat{x}_0)$ gives the target-timestep state $z_s = R_{t\to s}(z_t, \hat{x}_0)$. The baseline cache is an approximation because step-interval differences can cause small mismatch.

PEEK Prefix Cache. After selecting γ_i, the state at t_{i+1} is independent of future gates. We cache it, so a later candidate at position j starts from the cached state after step $j-1$ and computes only the suffix.

Search Cost. Let candidate index $i = 0, \ldots, N-2$, so its NFE-matched budget is $N+i+1$ in the worst case where every preceding gate is nonzero. Without caching, the initial partial schedule costs N queries, and both the ordinary baseline and the PEEK candidate are rerun at every budget:

$$C_n = N + 2\sum_{i=0}^{N-2}(N+i+1) = 3N^2 - 2N. \tag{12}$$

With caching, the initial partial schedule is evaluated once at a cost of N queries. The first $N+1$-query ordinary baseline is then evaluated in full. At each later position $i = 1, \ldots, N-2$, rebasing converts the cached pre-final state to the new grid, leaving only the new final-step query. Thus, the baseline cost is

$$C_b = N + 1 + \sum_{i=1}^{N-2} 1 = 2N - 1. \tag{13}$$



Figure 3: Qualitative Comparison. PEEK uses clean-prediction interpolation and generally achieves better detail-rendering or shape coherence for the generated images. Best viewed in color.



Figure 4: PEEK-G Qualitative Comparison. PEEK-G uses solver-direction interpolation and generally achieves better detail-rendering or shape coherence for the generated images. Best viewed in color.

For the PEEK candidate at position i, its cached prefix is free; one new base query, one new look-ahead query, and $N-1-i$ ordinary suffix queries remain. Hence

$$C_p = \sum_{i=0}^{N-2} [2 + (N-1-i)] = \frac{N^2+3N-4}{2}. \tag{14}$$

Including the initial partial schedule, the cached total is therefore $C_c = N + C_b + C_p = (N^2 + 9N - 6)/2$. For $N = 9$, the naive and cached counts are 225 and 78, respectively, a 65.3% reduction. Appendix E gives the expanded count.

4 Experiments

Evaluation Protocol. We evaluate unconditional pixel-space generation on CIFAR-10 [Krizhevsky, 2009] and LSUN Bedroom [Yu et al., 2015], class-conditional pixel-space generation on ImageNet at 64×64, class-conditional latent generation on ImageNet at 256×256 [Deng et al., 2009], and text-conditioned latent generation on MS-COCO [Lin et al., 2014]. For CIFAR-10 (Models A and B), LSUN Bedroom (basic and EDM), and ImageNet (64×64 and 256×256), we evaluate two models per dataset. We report Fréchet Inception Distance (FID) [Heusel et al., 2017], where lower is better. All methods in a comparison use the same model and evaluation protocol unless stated otherwise. Parenthetical percentages in the result tables report relative FID reductions over the corresponding NFE-matched baseline and are shown only when they exceed 5%.

Baselines. The NFE-matched baseline uses the same solver with additional ordinary steps so that it spends the same number of denoiser evaluations as PEEK. The NFE-matched baseline uses the same default schedule as PEEK's to ensure fairness. We compare DPM-Solver++, DEIS, Heun's 2nd and UniPC through the solver wrappers used by Zhou et al. [2023], Chen et al. [2024]. Heun's 2nd and

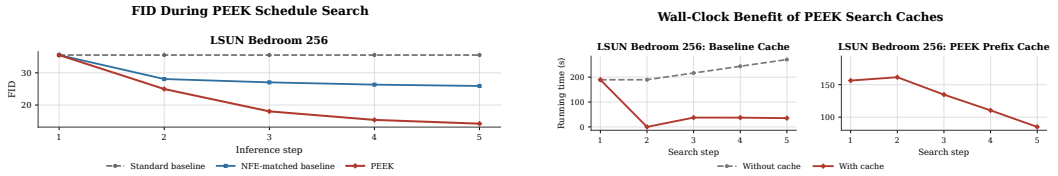

Figure 5: Schedule Search and Cache Behavior. The PEEK curve improves as useful interpolation positions are accepted. Prefix reuse and baseline rebasing reduce repeated denoising work during schedule search, especially as the selection moves toward later timesteps.

Method	LSUN FID (Base NFE 6) ↓	Recall ↑	Coverage ↑
DEIS, NFE-Matched	20.57	0.521	0.558
DPM-Solver++, NFE-Matched	15.95	0.567	0.656
UniPC, NFE-Matched	12.11	0.601	0.773
Heun's 2nd, NFE-Matched	51.69	0.369	0.254
Restart-style repeat	30.97	0.330	0.400
PEEK, DEIS	**11.38** (44.7%)↓	0.614	0.744
PEEK, DPM-Solver++	11.59 (27.3%)↓	**0.627**	**0.755**

Table 1: PEEK main results on the basic LSUN model [Yu et al., 2015]. Lower FID is better, while higher recall and coverage are better.

UniPC are included as explicit predictor-corrector baselines. We also include a Restart-style repeat baseline in Restart sampling [Xu et al., 2023]. Note that the original Restart sampling requires tuning of intervals and denoising repetition steps, but the exact method of tuning is underspecified. Therefore, we compare the Restart correction mechanism against ours using the exact same configurations (e.g., NFE), so as to keep variables controlled and contrast the core replacement mechanisms directly. The appendix specifies this baseline in more detail.

Models and Compute. For CIFAR-10 and the basic LSUN Bedroom model we use pretrained DDPM-family pixel-space models [Ho et al., 2020]. CIFAR-10 is evaluated at 32×32; LSUN Bedroom at 256×256; ImageNet at 64×64 or 256×256, depending on the model; and MS-COCO at 512×512. For ImageNet we use a DiT-XL/2-256 latent diffusion model with classifier-free guidance [Peebles and Xie, 2023, Ho and Salimans, 2021]. For MS-COCO we use a Stable Diffusion v1.5 latent model with the conventional text-guided evaluation configuration [Rombach et al., 2022]. We additionally evaluate ImageNet-EDM (64×64) and the EDM LSUN model (256×256), both based on the EDM training paradigm [Karras et al., 2022]; for the EDM LSUN model, we include iPNDM [Liu et al., 2022]. Schedule search and final sampling were run on multi-GPU H200 nodes. Pixel-space PEEK applies the interpolation in image space; latent-space PEEK applies the same clean-prediction interpolation in latent space before decoding for evaluation. Unless otherwise stated, all PEEK and PEEK-G visual samples use DPM-Solver++ schedules and sampling. In schedule search, vanilla PEEK uses gate candidates $\{0, 1\}$, while PEEK-G uses $\{0, 0.25, 0.5, 0.75, 1\}$.

PEEK Results. Table 1 shows that PEEK improves over strong NFE-matched baselines. With DEIS, FID decreases from 20.57 to 11.38 on the basic LSUN model, a 44.7% improvement. With DPM-Solver++, PEEK decreases FID from 15.95 to 11.59, a 27.3% improvement. The Restart-style repeat baseline under the same NFE budget is worse in these runs, suggesting our comparative advantage in the core exposure bias correction mechanism. Both PEEK and Restart-style repeat spend extra evaluations on correction rather than model retraining, but the correction signal is different: PEEK remains on the deterministic trajectory and edits the current clean prediction, while the Restart-style baseline revisits a noised state for stochastic re-noising and denoising. Heun's 2nd performs worse than the DPM-Solver++, DEIS, and UniPC baselines in these low-NFE settings, aligning with related reports and demonstrating our comparative advantage over this traditional predictor-corrector mechanism [Wang et al., 2025a].

(a) Additional Dataset FID

Dataset	Base NFE	NFE-Matched FID ↓	PEEK FID ↓
CIFAR-10 32×32 (Model A, 9 steps)	9	25.80	**20.23** (21.6%)↓
CIFAR-10 32×32 (Model A, 20 steps)	20	18.16	**15.05** (17.1%)↓
CIFAR-10 32×32 (Model B, 9 steps)	9	19.87	**16.46** (17.2%)↓
CIFAR-10 32×32 (Model B, 20 steps)	20	12.23	**10.04** (17.9%)↓
ImageNet 256×256	9	20.40	**19.64**
MS-COCO 512×512 (order 1)	9	14.58	**12.93** (11.3%)↓
MS-COCO 512×512 (order 3)	9	13.54	**11.74** (13.3%)↓

(b) Proxy Search

Dataset	Base NFE	256-Dim Proxy FID ↓	Search Time / Peak CUDA Memory	Final FID ↓
ImageNet 256×256	9	5.15	1.5 min / 39.0 GB	19.64
MS-COCO 512×512	9	2.79	2.2 min / 102.2 GB	12.93

Table 2: Additional datasets and proxy-search results with PEEK and DPM-Solver++ [Lu et al., 2022b].

Dataset / model	Resolution	Solver	Base NFE	NFE-Matched FID ↓	PEEK FID ↓
ImageNet-EDM	64×64	DPM-Solver++	6	19.71	**9.99** (49.3%)↓
ImageNet-EDM	64×64	DEIS	6	10.13	**7.29** (28.0%)↓
LSUN-Bedroom-EDM	256×256	DPM-Solver++	5	22.89	**11.77** (48.6%)↓
LSUN-Bedroom-EDM	256×256	DEIS	5	12.10	**8.79** (27.4%)↓
LSUN-Bedroom-EDM	256×256	iPNDM	5	15.86	**10.69** (32.6%)↓

Table 3: PEEK vs the NFE-matched baselines on EDM-trained models.

The NFE-matched rows are the intentionally compute-matched baselines. Improvements over these rows therefore indicate that PEEK is improving the use of computation. The PRDC results support the FID comparison. Relative to their NFE-matched baselines, DEIS PEEK raises recall from 0.521 to 0.614 and coverage from 0.558 to 0.744, while DPM-Solver++ PEEK raises recall from 0.567 to 0.627 and coverage from 0.656 to 0.755.

Table 2 reports additional datasets. On CIFAR-10, schedules are selected with the proxy FID and then rerun for standard final FID. To assess the sensitivity of PEEK to model choice, we test CIFAR-10 Models A and B, which are trained on the same dataset but have different generation quality. With CIFAR-10 Model A, a 9-step base solver reduces FID from 25.80 to 20.23, a 21.6% improvement. Increasing the base solver to 20 steps lowers both FID values, moving the NFE-matched baseline to 18.16 and PEEK to 15.05, a 17.1% improvement. With CIFAR-10 Model B, the same study gives lower absolute FID values: PEEK reduces 9-step FID from 19.87 to 16.46 (17.2%), and 20-step FID from 12.23 to 10.04 (17.9%).

On ImageNet, the binary proxy-selected PEEK schedule reduces FID from 20.40 to 19.64. On MS-COCO, the same proxy-search protocol reduces first-order DPM-Solver++ FID from 14.58 to 12.93, an 11.3% relative reduction. These results suggest that one-step-look-ahead correction can apply beyond one resolution, architecture family, or conditioning mode, but the effectiveness varies across models and variants. Note that the ImageNet result tests a class-conditional latent transformer, while MS-COCO tests text-guided latent sampling with a different prompt-conditioned pipeline.

Table 3 adds checkpoints trained under the EDM paradigm [Karras et al., 2022]. On ImageNet-EDM at 64×64, PEEK reduces FID by 49.3% for first-order DPM-Solver++ and 28.0% for first-order DEIS under the same compute. On the EDM LSUN model, the corresponding improvements are 48.6% for DPM-Solver++, 27.4% for DEIS, and 32.6% for iPNDM [Liu et al., 2022]. These EDM LSUN rows use a different EDM-trained model from the basic LSUN model in Table 1, so the two LSUN blocks should be read as separate models.

Dataset	Base NFE	NFE-Matched FID ↓	PEEK-G FID ↓
CIFAR-10 (Model B)	9	17.48	**13.18** (24.6%)↓
Basic LSUN model	6	15.95	**15.60**
ImageNet	9	20.40	**19.68**
MS-COCO	9	14.41	**10.66** (26.0%)↓

Table 4: PEEK-G versus the NFE-matched DPM-Solver++ baseline.

Method/γ	NFE	DPM-Solver++	DEIS
PEEK 01111 (Greedy Search Schedule)	10	**11.59**	**11.38**
PEEK 10111	10	12.63	13.34
PEEK 11011	10	13.82	14.39
PEEK 11101	10	13.50	13.51
PEEK 11110	10	13.90	13.36
NFE-Matched Baseline	11	14.43	18.45
PEEK 11111	11	12.89	12.98

Table 5: Basic LSUN Bedroom model schedule ablation.

Dataset / model	Base NFE	Total NFE	NFE-Matched FID-5K ↓	All-One PEEK FID-5K ↓
CIFAR-10 (Model A)	9	17	34.21	**29.15** (14.8%)↓
ImageNet-EDM 64×64	6	11	20.93	**15.05** (28.1%)↓
LSUN-Bedroom-EDM 256×256	5	9	21.59	**15.50** (28.2%)↓

Table 6: Fixed all-one PEEK schedules without schedule search.

Higher-Order Solver. We also evaluate third-order DPM-Solver++ on MS-COCO with the same binary interpolation schedule. The higher-order NFE-matched baseline improves over the first-order baseline from 14.58 to 13.54 FID, and PEEK further improves the higher-order run to 11.74 FID, a 13.3% relative reduction over its NFE-matched baseline, suggesting compatibility.

Schedule Effectiveness. Figure 5 shows representative search traces. The selected schedules are not uniformly dense. Interpolating at all peekable positions does not worsen the ordinary prediction on the basic LSUN Bedroom model. However, not every extra denoiser call is equally valuable, and very late interpolations can have limited impact because much of the layout has already been determined, while the early extra denoiser call may remain low-quality and insufficient to beat the NFE-matched baseline at one extra inference step. Therefore, the early extra denoiser call may also be skipped. Table 5 tests this directly on the basic LSUN Bedroom model by holding the number of interpolations fixed and changing only which candidate position is disabled; binary strings list the five peekable positions from left to right. It also reports the all-one schedule and its 11-step ordinary baseline. In this single-evaluation ablation, the all-one schedule improves its baseline but yields a higher FID than the searched schedule. All-one interpolation improves over its own NFE-matched 11-step baseline by 10.7% for DPM-Solver++ and 29.6% for DEIS. The searched schedule also uses fewer interpolation calls than the all-one schedule.

Table 6 evaluates the fixed all-one schedule more broadly, without schedule search, on DPM-Solver++. It improves over the NFE-matched baseline by 28.1% on ImageNet-EDM, and 28.2% on the EDM LSUN model. The all-one schedule requires no search and stills provide non-trivial generation-quality gains, demonstrating PEEK effectiveness at almost zero overhead. Nevertheless, the schedule search remains useful for selecting a sparser, higher-quality schedule.

Cache Effectiveness. Figure 5 summarizes the runtime effect of the cache strategy. The main gain comes from avoiding repeated prefixes: once the search has accepted the first several interpolation gates, testing a later candidate only requires denoising the remaining suffix.

PEEK-G Extension. Table 4 reports PEEK-G results on CIFAR-10 Model B, the basic LSUN model, ImageNet, and MS-COCO. It reduces FID by 24.6% on CIFAR-10 Model B and 26.0% on MS-COCO; the corresponding reductions on the basic LSUN model and ImageNet are 2.2% and 3.5%, respectively.

Qualitative Comparison. Figure 3 compares standard, NFE-matched, Restart-style repeat, and PEEK samples. PEEK tends to preserve the deterministic trajectory while sharpening object details. In contrast, Restart-style repeat can introduce larger stochastic deviations because it explicitly adds fresh noise before denoising. Figure 4 compares the NFE-matched samples against those generated by PEEK-G, which renders clearer details. The appendix includes additional qualitative samples.

5 Conclusion

We presented PEEK, a training-free and model-agnostic algorithm for improving deterministic diffusion sampling. By interpolating the current clean prediction with its look-ahead counterpart at searched positions to construct the next noisy state, PEEK improves the NFE-matched baselines in diverse image generation settings. The cache design makes schedule calibration more efficient. A current limitation is the drop in coverage, which could be mitigated as future work. Overall, PEEK spends inference computation on generally cleaner signals, potentially improving generation across different models and samplers, without model fine-tuning and or deployment overhead beyond the per-model calibration cost.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work was provided in part by the Ministry of Education, Singapore, under its MOE AcRF Tier 3 Grant (MOE-MOET32022-0001). The authors declare no competing interests.

References

Yuren Cai, Guangyi Wang, Zongqing Li, Li Li, Zhihui Liu, and Songzhi Su. Three creates all: You only sample 3 steps. *arXiv preprint arXiv:2603.22375*, 2026.

Defang Chen, Zhenyu Zhou, Jian-Ping Mei, Chunhua Shen, Chun Chen, and Can Wang. A geometric perspective on diffusion models. *arXiv preprint arXiv:2305.19947*, 2023.

Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ode-based diffusion sampling. *arXiv preprint arXiv:2405.11326*, 2024.

Pedro Cisneros-Velarde. It takes two to tango: Two parallel samplers improve quality in diffusion models for limited steps. *arXiv preprint arXiv:2510.21802*, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-order denoising diffusion solvers. In *Advances in Neural Information Processing Systems*, 2022.

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 *Advances in Neural Information Processing Systems*, 2017.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Sangwoo Jo and Sungjoon Choi. Formalizing the sampling design space of diffusion-based generative models via adaptive solvers and wasserstein-bounded timesteps. *arXiv preprint arXiv:2602.12624*, 2026.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Inho Kong, Sojin Lee, Youngjoon Hong, and Hyunwoo J. Kim. Error as signal: Stiffness-aware diffusion sampling via embedded runge-kutta guidance. *arXiv preprint arXiv:2603.03692*, 2026.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Lijiang Li, Huixia Li, Xiawu Zheng, Jie Wu, Xuefeng Xiao, Rui Wang, Min Zheng, Xin Pan, Fei Chao, and Rongrong Ji. AutoDiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. *arXiv preprint arXiv:2305.08891*, 2023.

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 *European Conference on Computer Vision*, 2014.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

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. In *Advances in Neural Information Processing Systems*, 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.

Byeonghu Na, Yeongmin Kim, Minsang Park, Donghyeok Shin, Wanmo Kang, and Il-chul Moon. Diffusion rejection sampling. In *International Conference on Machine Learning*, 2024.

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. *arXiv preprint arXiv:2308.15321*, 2023.

Aleksandr Oganov, Ilya Bykov, Eva Neudachina, Mishan Aliev, Alexander Tolmachev, Alexander Sidorov, Aleksandr Zuev, Andrey Okhotin, Denis Rakitin, and Aibek Alanov. GAS: Improving discretization of diffusion ODEs via generalized adversarial solver. *arXiv preprint arXiv:2510.17699*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Jianning Pei, Han Hu, and Shuyang Gu. Optimal stepsize for diffusion sampling. *arXiv preprint arXiv:2503.21774*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. *arXiv preprint arXiv:2404.14507*, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

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.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Guangyi Wang, Wei Peng, Lijiang Li, Wenyu Chen, Yuren Cai, and Songzhi Su. Diffusion sampling correction via approximately 10 parameters. In *International Conference on Machine Learning*, 2025a.

Ruoyu Wang, Beier Zhu, Junzhi Li, Liangyu Yuan, and Chi Zhang. Adaptive stochastic coefficients for accelerating diffusion sampling. In *Advances in Neural Information Processing Systems*, 2025b.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. In *Advances in Neural Information Processing Systems*, 2023.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. In *arXiv preprint arXiv:1506.03365*, 2015.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. DPM-Solver-v3: Improved diffusion ODE solver with empirical model statistics. In *Advances in Neural Information Processing Systems*, 2023.

Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. *arXiv preprint arXiv:2312.00094*, 2023.

A Diffusion and Sampling Preliminaries

This appendix briefly reviews the diffusion notation used in the main text and clarifies how PEEK fits into standard deterministic sampling. We focus on the variance-preserving notation used in our cache derivation and on the clean-prediction interface used by the samplers in our experiments.

Forward Noising Process. Diffusion models define a family of noisy marginals by gradually corrupting a clean data sample x_0. Under the standard variance-preserving parameterization, the noisy variable at diffusion time t can be written as

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

or equivalently

$$q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I\big) .$$

Here $\bar{\alpha}_t$ is the cumulative signal coefficient. Large diffusion times correspond to high-noise states, where $\bar{\alpha}_t$ is small and most clean-image structure has been obscured. Small diffusion times correspond to low-noise states, where the clean image is more visible.

The marginal distribution of forward-noised real data at time t is denoted by q_t. During training, the denoising network is supervised on samples $x_t \sim q_t$, so its inputs are noisy versions of real data rather than states produced by its own previous predictions.

Clean-Prediction Notation. Denoising networks can be parameterized in several equivalent ways, such as predicting noise, predicting the clean sample, or predicting a velocity variable. PEEK operates on the clean prediction, so throughout the paper we write

$$\hat{x}_{0,i} = D_\theta(z_i, t_i)$$

for the clean-sample estimate obtained from the denoiser at sampler state z_i and timestep t_i. If the underlying model predicts noise $\epsilon_\theta(x_t, t)$, this clean estimate is obtained by the usual conversion

$$D_\theta(x_t, t) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}.$$

Other prediction parameterizations are converted analogously. Thus D_θ should be read as the denoiser after conversion to a clean-sample estimate, not necessarily as the model's native output.

The subscript convention is important. The subscript 0 in x_0 and $\hat{x}_{0,i}$ denotes the clean endpoint of the diffusion process. We use z_i for the noisy sampler state at discrete index i and timestep t_i; $\hat{x}_{0,i}$ is the clean prediction made from that state. This distinction avoids overloading x_0 as both the clean endpoint and the initial high-noise sampler state.

Deterministic Sampling. At inference time, a deterministic sampler starts from a high-noise state and follows a decreasing timestep grid

$$t_0 > t_1 > \cdots > t_{N-1}.$$

Let z_i denote the sampler state at timestep t_i. A first-order deterministic solver step can be written abstractly as

$$z_{i+1} = \Phi_i(z_i, \hat{x}_{0,i}), \qquad \hat{x}_{0,i} = D_\theta(z_i, t_i),$$

where Φ_i includes the sampler-specific conversion between the clean prediction, the implied noise prediction, and the next noisy state. This notation covers the deterministic first-order wrappers used in our experiments.

For example, a DDIM-style update under the same variance-preserving parameterization first computes the implied noise

$$\hat{\epsilon}_i = \frac{z_i - \sqrt{\bar{\alpha}_{t_i}}\hat{x}_{0,i}}{\sqrt{1 - \bar{\alpha}_{t_i}}},$$

and then forms the next state by reusing this clean/noise decomposition at the lower-noise timestep:

$$z_{i+1} = \sqrt{\bar{\alpha}_{t_{i+1}}}\hat{x}_{0,i} + \sqrt{1 - \bar{\alpha}_{t_{i+1}}}\hat{\epsilon}_i.$$

Different deterministic solvers use different numerical updates, but they share the same practical structure: the denoiser is queried at the current state, the prediction is converted into the sampler's preferred parameterization, and the next state is produced without adding fresh random noise.

DPM-Solver Direction Form. DPM-Solver-style samplers write the reverse diffusion dynamics as an ODE in a noise-time variable and approximate the integral of a model-derived direction over each solver interval [Lu et al., 2022a,b]. Abstractly, the update can be written as

$$z_{i+1} = A_i z_i + \sum_{j=0}^{r-1} B_{i,j} d_{i-j}, \qquad d_i = G_i(z_i, D_\theta(z_i, t_i)),$$

where the coefficients A_i and $B_{i,j}$ are determined by the timestep grid and solver order, and d_i is the current model direction after converting the denoiser output into the solver's parameterization. First-order DPM-Solver++ uses only the current direction; multistep variants additionally use stored directions from previous timesteps.

A.1 PEEK-G Formalization

PEEK interpolates the clean prediction before the next noisy state is formed. For DPM-Solver-style updates, the same look-ahead information can also be used at the solver-direction level. Let

$$\hat{x}_{0,i}^{\text{base}} = D_\theta(z_i, t_i), \qquad d_i^{\text{base}} = G_i(z_i, \hat{x}_{0,i}^{\text{base}}) \tag{15}$$

denote the model direction used by the deterministic solver at timestep t_i, where G_i is the sampler-specific conversion from the denoiser output to the ODE direction. At a nonzero gate, PEEK-G first forms the same provisional next state $\tilde{z}_{i+1}$ as in PEEK using the ordinary scheduler step, but does not commit that state. It then evaluates the denoiser at the look-ahead state,

$$\hat{x}_{0,i}^{\text{look}} = D_\theta(\tilde{z}_{i+1}, t_{i+1}), \qquad d_i^{\text{look}} = G_{i+1}(\tilde{z}_{i+1}, \hat{x}_{0,i}^{\text{look}}). \tag{16}$$

For a nonzero gate, the selected solver direction is

$$d_i^{\text{sel}} = (1 - \gamma_i) d_i^{\text{base}} + \gamma_i d_i^{\text{look}}. \tag{17}$$

For a zero gate, PEEK-G instead sets $d_i^{\text{sel}} = d_i^{\text{base}}$ without computing d_i^{look}. The actual DPM-Solver step is still applied from the current state z_i with the usual timestep coefficients for the current solver interval. Fractional gates are natural in this direction space, so our PEEK-G searches use $\gamma_i \in \{0, 0.25, 0.5, 0.75, 1\}$. The solver coefficients remain those of the current interval, so PEEK-G is not a new DPM-Solver derivation. It is a post-hoc direction correction using a model prediction from a higher-SNR state. This is mathematically coherent with the diffusion target because denoiser predictions at different timesteps are trained toward the same clean data endpoint x_0, after the appropriate parameterization conversion. The practical risk is that the look-ahead prediction may be confidently biased; in that case PEEK-G can amplify the bias even while sharpening the local direction.

NFE Accounting. We count one function evaluation for each call to D_θ. A base deterministic trajectory with N denoiser queries therefore uses N NFEs. PEEK can be applied at the first $N - 1$ positions, because the final denoising query has no later noisy timestep inside the trajectory from which to obtain a look-ahead prediction.

If K interpolation gates in the PEEK schedule are nonzero, then PEEK makes K additional look-ahead denoiser calls. Its deployment cost is therefore $N + K$ NFEs. The NFE-matched baseline uses the same solver with $N + K$ ordinary denoiser evaluations and no look-ahead interpolation.

Training Distribution versus Sampling Trajectory. A useful way to view low-NFE sampling is through the mismatch between the training inputs and the inference-time states. During training, the denoiser is optimized on $x_t \sim q_t$, where q_t is obtained by forward noising real data. During deterministic sampling, however, the state z_i is generated recursively by the sampler and the model's own previous predictions. Let p_i denote the marginal distribution of these generated states at timestep t_i. In general,

$$p_i \neq q_{t_i}.$$

This mismatch is especially important in the low-NFE regime, where each solver step is large and early prediction errors can move the trajectory away from the forward-noised data manifold.

Under squared clean-prediction loss, the ideal training-time predictor at timestep t_i is the conditional mean

$$m_i(x) = \mathbb{E}_q[x_0 \mid x_{t_i} = x].$$

The denoiser D_θ is trained to approximate this target on samples from q_{t_i}. At inference time, however, it is queried on samples from p_i. When p_i contains states that are less data-like than the training marginal, the clean prediction $\hat{x}_{0,i}$ can be biased or unstable. This motivates using information from a nearby later state on the same deterministic trajectory.

Pixel-Space and Latent-Space Usage. The same notation applies to both pixel-space and latent-space diffusion models. For pixel-space models, z_i and $\hat{x}_{0,i}$ are image-space tensors. For latent diffusion models, they are latent tensors, and decoding is applied only after sampling for evaluation or visualization. In both cases, PEEK modifies the predicted clean sample in the space where the deterministic solver operates.

B Further Implementation Details

Solvers and Timestep Grids. Most reported PEEK runs use deterministic first-order solver wrappers to keep the comparison variables controlled. The DPM-Solver++ rows use the dpmsolver++ algorithm type in the scheduler implementation [Lu et al., 2022b]; the MS-COCO higher-order check additionally uses the order-3 DPM-Solver++ setting. DEIS rows use the DEIS update rule [Zhang and Chen, 2023] but, unless explicitly marked otherwise, reuse the DPM-Solver++ binary interpolation schedule for the same dataset and base step count. This makes the DEIS result a transfer test of the learned correction positions while preserving the same $N + K$ denoiser-evaluation budget as its NFE-matched DEIS baseline.

Models, Resolution, and Conditioning. CIFAR-10 Model A uses the google/ddpm-cifar10-32 checkpoint evaluated at 32^2 resolution, and CIFAR-10 Model B uses the DanielBairamian/ddpm-cifar10-32-ema checkpoint for the second CIFAR study and PEEK-G CIFAR rows. Both checkpoints expose the same UNet2DModel macro-architecture in Diffusers: 35.75M parameters, block widths $(128, 256, 256, 256)$, two residual layers per block, and one attention block at the 16-by-16 scale. The basic LSUN model uses google/ddpm-ema-bedroom-256. The EDM table additionally uses the ImageNet 64^2 conditional EDM checkpoint and the EDM LSUN model checkpoint at 256^2 [Karras et al., 2022]. For the EDM LSUN model, we include an iPNDM baseline [Liu et al., 2022]. ImageNet uses facebook/DiT-XL-2-256 in latent space with classifier-free guidance scale 4.0. MS-COCO uses a Stable Diffusion v1.5 checkpoint fine-tuned/evaluated with guidance scale 7.5, discrete latent-diffusion timesteps, and 512×512 decoding before FID evaluation. The latent-space experiments interpolate the predicted clean latent, not decoded RGB images.

Distributed Evaluation. The pixel-space and latent-space evaluations use multi-GPU H200 nodes with one sample shard per process. The CIFAR-10 qualitative grid uses saved snapshots from the 9-step proxy runs.

Interpolation in Pixel and Latent Spaces. For pixel-space DDPM models, the clean prediction in the main-paper PEEK interpolation equation is an image-space prediction. For latent diffusion models, the same operation is applied to the predicted latent clean sample. FID and qualitative visualizations are computed after the standard decoder is applied.

Candidate Gates. Vanilla PEEK uses binary interpolation gates $\{0, 1\}$. PEEK-G uses the direction-space candidate set $\{0, 0.25, 0.5, 0.75, 1\}$.

NFE-Matched Baseline. If an N-step PEEK schedule uses K nonzero look-ahead positions, then the deployment run uses $N + K$ denoiser evaluations. The NFE-matched baseline is the same deterministic solver run with $N + K$ ordinary solver steps and no look-ahead interpolation.

Restart-Style Repeat Baseline. The Restart-style baseline follows the same NFE budget as the corresponding PEEK run. After each selected main step, it forwards the current state from the next lower-noise timestep back to the previous higher-noise timestep by adding fresh Gaussian noise, then runs one deterministic solver step back to the lower-noise timestep. The selected repeat positions are the nonzero entries of the PEEK schedule, so a schedule with K nonzero entries also spends K extra denoiser evaluations [Xu et al., 2023].

Dataset	Evaluation sampler	Base NFE	Total NFE	γ
PEEK				
Unconditional Pixel-Space Generation on CIFAR-10	DPM-Solver++	9	17	$(1, 1, 1, 1, 1, 1, 1, 1, 1)$
Unconditional Pixel-Space Generation on CIFAR-10 (EMA, PEEK)	DPM-Solver++	9	11	$(0, 0, 0, 0, 0, 0, 0, 1, 1)$
Unconditional Pixel-Space Generation on CIFAR-10 (EMA, PEEK)	DPM-Solver++	20	23	$(0, 0, 0, 0, 0, 0, 0, 0,$ $0, 0, 0, 0, 0, 0, 0, 0,$ $0, 0, 1, 1, 1)$
Unconditional Pixel-Space Generation on Basic LSUN Bedroom Model	DPM-Solver++ / DEIS	6	10	$(0, 1, 1, 1, 1)$
Class-Conditional Latent Generation on ImageNet 256×256	DPM-Solver++	9	13	$(0, 0, 0, 0, 0, 1, 1, 1, 1)$
Text-Conditional Latent Generation on MS-COCO	DPM-Solver++ (order 1 / 3)	9	12	$(0, 0, 0, 0, 0, 0, 1, 1, 1)$
Conditional EDM Generation on ImageNet 64×64	DPM-Solver++ / DEIS	6	9	$(1, 1, 1, 0, 0)$
Unconditional EDM Generation on EDM LSUN Model	DPM-Solver++ / DEIS / iPNDM	5	8	$(1, 1, 1, 0)$
PEEK-G				
Unconditional Pixel-Space Generation on CIFAR-10 (EMA, PEEK-G)	DPM-Solver++	9	13	$(1, 0, 0, 0, 0, 1, 1, 1, 1)$
Unconditional Pixel-Space Generation on Basic LSUN Model (PEEK-G)	DPM-Solver++	6	10	$(0, 0.5, 1, 0.5, 0.5)$
Class-Conditional Latent Generation on ImageNet 256×256 (PEEK-G)	DPM-Solver++	9	13	$(0, 0, 0, 0, 0.5,$ $0.5, 0.25, 1)$
Text-Conditional Latent Generation on MS-COCO (PEEK-G)	DPM-Solver++ (order 1)	9	13	$(0, 0, 0, 0, 1,$ $1, 1, 0.25)$

Table 7: Interpolation schedules used in the reported final evaluations. NFE counts the ordinary solver calls plus nonzero look-ahead calls. Vanilla PEEK uses binary gates, while PEEK-G searches over $\{0, 0.25, 0.5, 0.75, 1\}$.

Evaluation Artifacts. Final FID values are measured on held-out samples distinct from those used for validation search. We use the StyleGAN Inception feature extractor for FID statistics, with 2048-dimensional features for reported final FID except where explicitly marked as a 256-dimensional proxy. CIFAR-10, ImageNet, and MS-COCO schedules are selected with lightweight proxy runs and then rerun with the fixed schedules under the standard final-reporting protocol. Unless otherwise stated, qualitative PEEK and PEEK-G figures use DPM-Solver++ schedules and samples.

Selected Schedules. Table 7 lists the schedules used for the main final evaluations. Each entry has one fewer interpolation gate than solver steps because the final denoising step has no later timestep to peek at. The DEIS final evaluation on the basic LSUN model uses the corresponding DPM-Solver++ schedule as a transfer test; the EDM DEIS and iPNDM rows likewise reuse the corresponding DPM-Solver++ schedule. All other rows use the schedule searched with the same sampler as the final evaluation.

Calibration Cost. Schedule calibration uses proxy metrics. On CIFAR-10, we use a 1,000-sample, 256-dimensional proxy for the DPM-Solver++ schedule sweep, then rerun the fixed schedules with the standard final-reporting protocol. The 9-step schedule search took 20.4 seconds, and the 20-step schedule search took 32.5 seconds; both used about 2.2 GB peak CPU resident set size and 8.7 GB peak CUDA reserved memory. For the latent ImageNet and MS-COCO runs, the main-paper additional-datasets table reports the proxy-search time and peak CUDA memory for PEEK before the selected schedules are rerun with the standard final evaluation metric. Once calibration is complete, the schedule is fixed and no search is run at deployment, so inference speed is determined only by the selected NFE budget.

C Extended Related Work

Training and Distillation for Fast Generation. One way to reduce few-step sampling error is to train or distill a model specifically for short trajectories. Progressive distillation compresses a many-step diffusion sampler into a smaller number of student steps [Salimans and Ho, 2022], while consistency models learn mappings that support direct or few-step generation [Song et al., 2023]. Recent lightweight variants optimize a small sampler or conditioning component instead of the whole denoiser: GAS distills and adversarially trains a generalized ODE solver [Oganov et al., 2025], and MTEO freezes the pretrained backbone while optimizing step-wise, layer-wise time embeddings. These approaches can deliver strong low-NFE deployment with little or no per-step overhead after training [Cai et al., 2026]. PEEK addresses a different operating point: it is training-free and can be applied to an existing sampler immediately, at the cost of selected extra denoiser calls during inference.

Sampler and Schedule Design. DDIM introduced deterministic non-Markovian sampling paths that can use fewer denoising steps than ancestral sampling [Song et al., 2021a]. PNDM, DPM-Solver, DPM-Solver++, DEIS, UniPC, GENIE, AMED-Solver, and DPM-Solver-v3 further improve low-NFE performance through ODE-aware integration, multistep history, exponential-integration formulas, learned directions, solver statistics, or predictor-corrector design [Liu et al., 2022, Lu et al., 2022a,b, Zhang and Chen, 2023, Zhao et al., 2023, Dockhorn et al., 2022, Zhou et al., 2023, Zheng et al., 2023]. Heun's method is a representative predictor-corrector update: it forms a predicted next state and combines the current and predicted ODE derivatives, whereas PEEK uses the predicted state only to obtain a clean-sample estimate that can interpolate the current clean prediction before the next noisy state is formed. EDM and schedule-alignment work show that noise parameterization, timestep placement, and architecture allocation are also central to sample quality [Karras et al., 2022, Li et al., 2023, Sabour et al., 2024, Chen et al., 2024, Pei et al., 2025]. SDM formalizes an adaptive sampling design that uses low-order solvers in early high-noise stages and progressively deploys higher-order solvers later [Jo and Choi, 2026]. ERK-Guid instead exploits embedded Runge-Kutta local-error estimates as a guidance signal, so its mechanism is specific to embedded/higher-order ODE-solver structure [Kong et al., 2026]. PEEK is orthogonal to these advances: it uses the same timestep grid at deployment and changes the denoiser-derived clean estimate inside selected deterministic steps.

Inference-Time Correction. Restart sampling alternates a backward ODE path with forward noising segments, using stochasticity to contract accumulated sampling error [Xu et al., 2023]. Among prior training-free inference-time methods, this is the closest in spirit because it spends extra denoiser evaluations on correction rather than only densifying an ordinary trajectory. The mechanism is different: Restart exposes the sampler to a fresh-noise sub-trajectory and requires interval boundaries, restart counts, and sub-trajectory lengths; PEEK uses a deterministic provisional next state only to obtain a one-step-later clean prediction, then applies the correction to the current step. Recent adaptive-stochasticity plugins make a different tradeoff. AdaSDE learns a per-step stochastic coefficient through lightweight distillation, dynamically regulating correction strength while remaining compatible with existing solvers [Wang et al., 2025b]. PEEK uses no distillation and no injected stochasticity; its correction is deterministic once the schedule is fixed, so it can be layered conceptually with stochastic or learned correction methods but occupies the simpler training-free setting. DiffRS is also post-hoc, but uses a learned time-dependent discriminator to accept or reject intermediate samples and therefore refines different samples with different effort [Na et al., 2024]. DiffRS is attractive when sample-adaptive quality filtering and auxiliary discriminator training are acceptable, whereas PEEK is preferable when a fixed deployment NFE budget, no extra learned model, and a simple deterministic trajectory are desired. SE2P is another relevant inference-time correction: it uses two parallel samplers at successive times and integrates information from the later-step processor into the earlier one without adding extra model calls per processor [Cisneros-Velarde, 2025]. This is conceptually close to PEEK's use of a later-step state as a correction signal, but the computational assumptions differ. SE2P relies on parallel processors, whereas PEEK uses explicit extra look-ahead evaluations within one sampler and reports comparisons against NFE-matched ordinary trajectories. Related analyses of diffusion trajectories and schedule flaws emphasize that sampler states encountered during inference can differ from idealized training or terminal distributions [Lin et al., 2023, Chen et al., 2023]. PEEK targets this issue without injecting fresh noise, changing the timestep grid, or deriving a new solver: the correction comes from the model's own next-step clean

prediction along the current deterministic trajectory and is selected with an automated metric-driven search.

D Pseudocode

PEEK Sampling. Given timesteps $t_0, \ldots, t_{N-1}$, initial state z_0, and schedule $\boldsymbol{\gamma}$:

1. For $i = 0, \ldots, N-2$, compute the ordinary prediction $\hat{x}_{0,i}^{\text{base}} = D_\theta(z_i, t_i)$.
2. If $\gamma_i = 0$, set $z_{i+1} = \Phi_i(z_i, \hat{x}_{0,i}^{\text{base}})$ and continue.
3. Otherwise compute the provisional state $\tilde{z}_{i+1} = \Phi_i(z_i, \hat{x}_{0,i}^{\text{base}})$.
4. Compute the look-ahead prediction $\hat{x}_{0,i}^{\text{look}} = D_\theta(\tilde{z}_{i+1}, t_{i+1})$.
5. Select $\hat{x}_{0,i}^{\text{sel}} = \hat{x}_{0,i}^{\text{look}}$.
6. Take the actual step $z_{i+1} = \Phi_i(z_i, \hat{x}_{0,i}^{\text{sel}})$.
7. Run the final ordinary solver step at t_{N-1}.

PEEK-G Sampling. PEEK-G follows the same provisional look-ahead procedure, but interpolates the solver direction rather than the clean prediction. At step i, compute the ordinary direction d_i^{base}. If $\gamma_i = 0$, take the ordinary update without a look-ahead query. Otherwise, form $\tilde{z}_{i+1}$, query the denoiser at $(\tilde{z}_{i+1}, t_{i+1})$, convert the result to d_i^{look}, and use $(1-\gamma_i)d_i^{\text{base}} + \gamma_i d_i^{\text{look}}$ inside the ordinary DPM-Solver update for the current interval. The final step remains an ordinary solver step because no later timestep is available.

DPM-Solver with PEEK-G. For a DPM-Solver-style sampler, let $U_i(z_i, d_i)$ denote the ordinary solver update over interval $[t_i, t_{i+1}]$ after the denoiser output has been converted to the solver direction d_i. Given a direction schedule $\boldsymbol{\gamma}$:

1. For $i = 0, \ldots, N-2$, evaluate the denoiser at the current state and form the ordinary direction $d_i^{\text{base}} = G_i(z_i, D_\theta(z_i, t_i))$.
2. If $\gamma_i = 0$, set $z_{i+1} = U_i(z_i, d_i^{\text{base}})$ and continue.
3. Take the provisional DPM-Solver step $\tilde{z}_{i+1} = U_i(z_i, d_i^{\text{base}})$.
4. Query the denoiser at the provisional next state, $\hat{x}_{0,i}^{\text{look}} = D_\theta(\tilde{z}_{i+1}, t_{i+1})$, and convert it to a look-ahead direction $d_i^{\text{look}} = G_{i+1}(\tilde{z}_{i+1}, \hat{x}_{0,i}^{\text{look}})$.
5. Blend the direction according to the selected gate:
$$d_i^{\text{sel}} = (1-\gamma_i)d_i^{\text{base}} + \gamma_i d_i^{\text{look}}.$$
6. Apply the ordinary DPM-Solver update with the original interval coefficients:
$$z_{i+1} = U_i(z_i, d_i^{\text{sel}}).$$
7. At t_{N-1}, run the final ordinary DPM-Solver step, because there is no later timestep from which to obtain a look-ahead direction.

Greedy Schedule Search. For each peekable position i, the search reuses the selected prefix state, evaluates candidate interpolation gates on the remaining suffix, compares the best candidate to the NFE-matched ordinary baseline, and stores the selected prefix state for position $i+1$. Vanilla PEEK uses $\mathcal{A} = \{0, 1\}$, while PEEK-G uses $\mathcal{A} = \{0, 0.25, 0.5, 0.75, 1\}$. The selected schedule is then used unchanged for final evaluation.

E Cache Derivation

Worst-Case Count. We count denoiser queries and index the $N-1$ peekable positions by $i = 0, \ldots, N-2$. In the worst case, every previous gate is nonzero, so the NFE-matched budget at position i is $N + i + 1$. Without caching, the initial current schedule costs N queries. The ordinary

baseline and the PEEK candidate are then both rerun in full at each budget from $N+1$ through $2N-1$. Thus

$$C_n = N + 2\sum_{i=0}^{N-2}(N+i+1). \tag{18}$$

Changing the summation index gives

$$C_n = N + 2\sum_{m=N+1}^{2N-1} m = 3N^2 - 2N. \tag{19}$$

The two caches change the count as follows. The initial partial schedule is evaluated once at a cost of N queries, and its metric supplies the zero-gate candidate at the first position. The first ordinary baseline costs $N+1$ queries. At each later position $i = 1, \ldots, N-2$, baseline rebasing maps the cached pre-final state to the new grid, leaving only the new final-step query. Consequently,

$$C_b = N + 1 + \sum_{i=1}^{N-2} 1 = 2N - 1. \tag{20}$$

For the PEEK candidate at position i, the selected prefix contains all earlier base and look-ahead results. Only one new base query, one new look-ahead query, and the $N-1-i$ ordinary queries in the remaining suffix must be evaluated. Therefore,

$$C_p = \sum_{i=0}^{N-2}\left[1 + 1 + (N-1-i)\right]. \tag{21}$$

Reversing the suffix index gives

$$C_p = 2(N-1) + \sum_{r=1}^{N-1} r = \frac{N^2+3N-4}{2}. \tag{22}$$

Including the initial partial schedule, the cached search cost is

$$C_c = N + C_b + C_p = \frac{N^2+9N-6}{2}. \tag{23}$$

For the paper's $N = 9$ search, the naive count is $9 + 2(10 + \cdots + 17) = 225$. The initial partial schedule costs 9, the cached baseline costs $10 + 7 \cdot 1 = 17$, and the cached PEEK candidates cost $2 \cdot 8 + (8 + \cdots + 1) = 52$, for a total of 78. The reduction is $(225 - 78)/225 = 65.3\%$. These sums count denoiser queries; cached base and look-ahead results incur no new query. Additional nonzero gate candidates add their corresponding PEEK candidate work.

F Example Search Log

step 0: NFE-matched baseline FID 3.0802; current PEEK FID 3.2585; candidate $\gamma = 1$ gives 3.2459; selected $\gamma = 0$.
step 5: NFE-matched baseline FID 3.0802; current PEEK FID 3.2585; candidate $\gamma = 1$ gives 3.0634; selected $\gamma = 1$.
step 7: NFE-matched baseline FID 2.9720; current PEEK FID 2.9003; candidate $\gamma = 1$ gives 2.7922; selected $\gamma = 1$.

G Reproducibility and Responsible Use

All reported results use pretrained image generators and image benchmarks. The method does not introduce new training data or a new model checkpoint; it is an inference procedure plus a schedule-calibration routine. Code and reproduction materials are available from the project website at https://dachengruoque.github.io/PEEK-Video/, including the schedules, commands, and evaluation utilities needed to reproduce the reported sampling results. Image generation systems can be misused to synthesize deceptive or harmful visual content. PEEK can improve generation quality, so releases should follow the same safeguards and usage restrictions as the underlying generative models.

Existing Assets. The experiments use benchmarks and pretrained generators: CIFAR-10 [Krizhevsky, 2009], LSUN Bedroom [Yu et al., 2015], ImageNet [Deng et al., 2009], MS-COCO [Lin et al., 2014], DDPM-family checkpoints [Ho et al., 2020], DiT [Peebles and Xie, 2023], Stable Diffusion [Rombach et al., 2022], and solver implementations following DPM-Solver++, DEIS, UniPC, and fast-sampler codebases [Lu et al., 2022b, Zhang and Chen, 2023, Zhao et al., 2023, Zhou et al., 2023, Chen et al., 2024].

H Additional Visualizations

Figures 6-10 provide additional PEEK samples, and Figures 11-19 provide PEEK-G samples and direct qualitative contrasts.



Figure 6: Basic LSUN Bedroom model PEEK Qualitative Comparison I. Samples are generated by PEEK with the google/ddpm-ema-bedroom-256 pixel-space DDPM checkpoint. Best viewed in color.



Figure 7: Basic LSUN Bedroom model PEEK Qualitative Comparison II. Samples are generated by PEEK with the google/ddpm-ema-bedroom-256 pixel-space DDPM checkpoint. Best viewed in color.



Figure 8: ImageNet PEEK Qualitative Comparison. Samples are generated by PEEK with the facebook/DiT-XL-2-256 class-conditional latent diffusion transformer with classifier-free guidance. Best viewed in color.

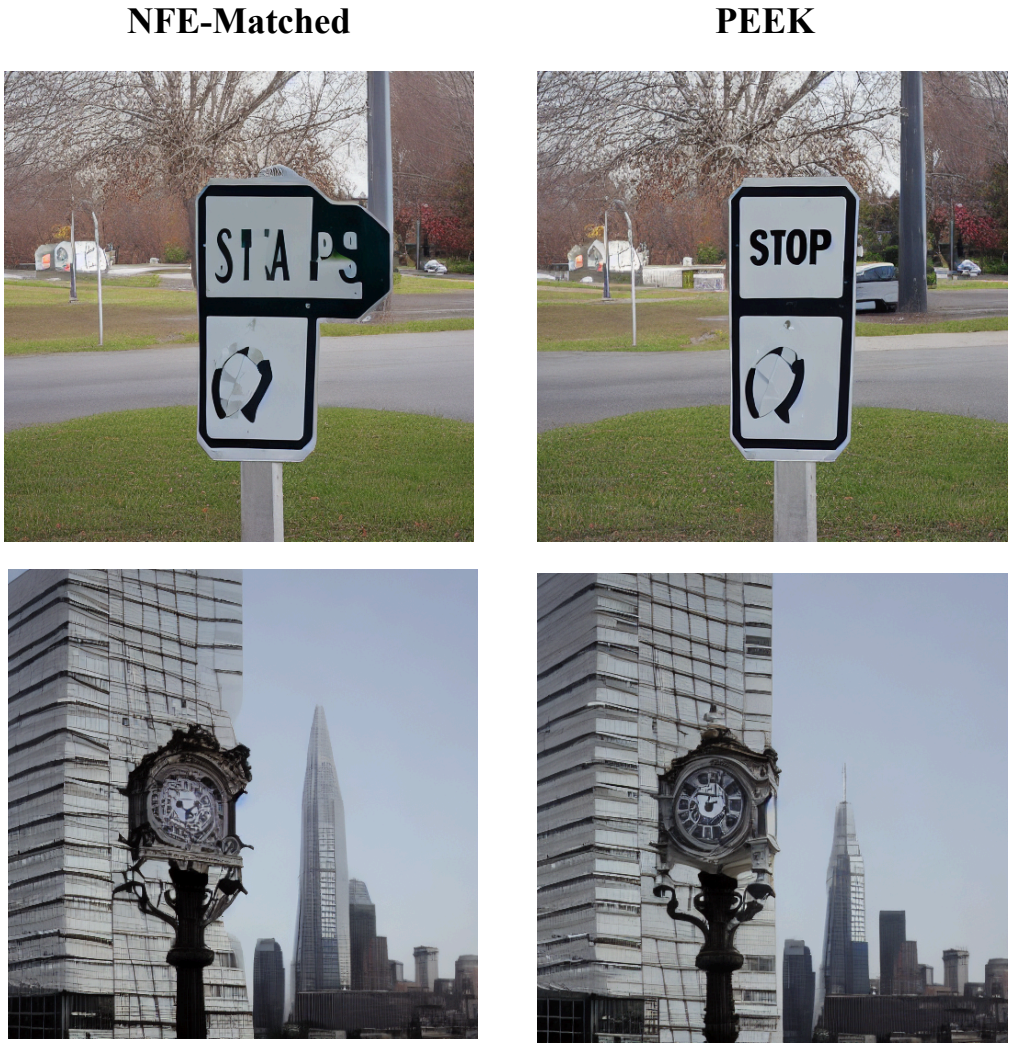

Figure 9: MS-COCO PEEK Qualitative Comparison I. Samples are generated by PEEK with the Stable Diffusion v1.5 latent text-to-image checkpoint. Best viewed in color.

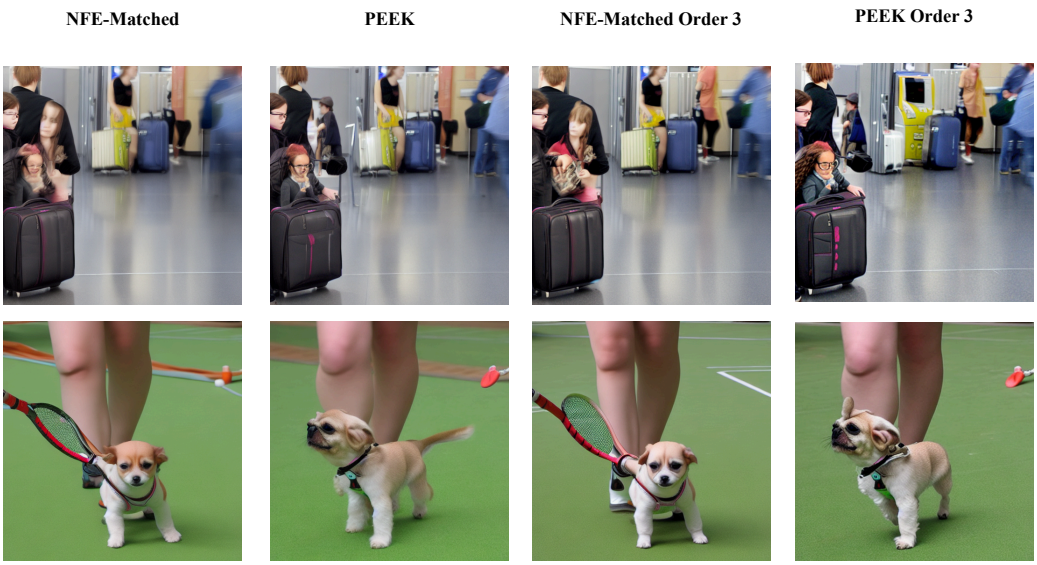

Figure 10: MS-COCO PEEK Qualitative Comparison II. Samples are generated by PEEK with the Stable Diffusion v1.5 latent text-to-image checkpoint. Best viewed in color.



Figure 11: CIFAR-10 PEEK-G Qualitative Comparison. Each method block contains five rows with 20 samples per row. PEEK-G is compared against the NFE-matched DPM-Solver++ baseline under the same denoiser-evaluation budget. Best viewed in color.

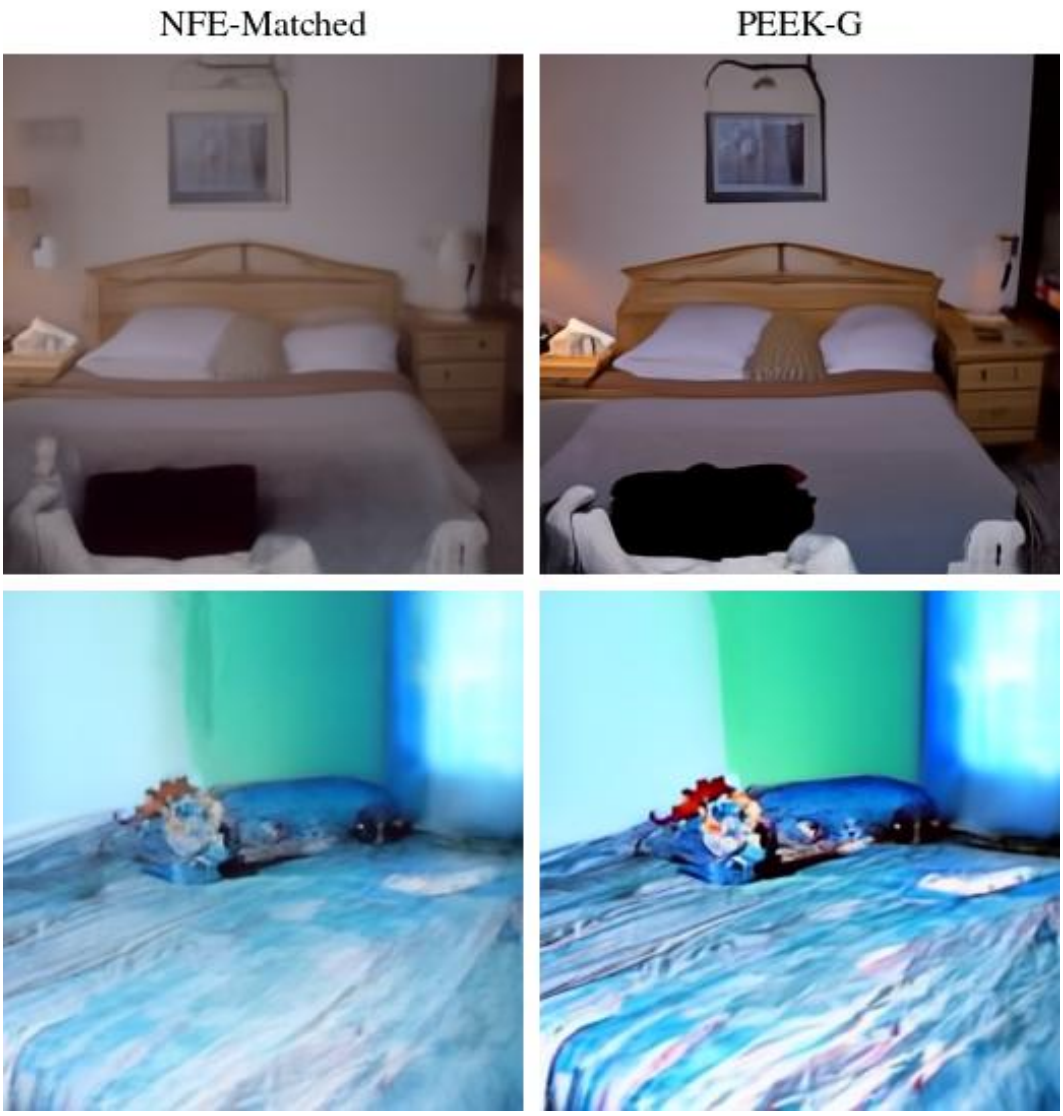

Figure 12: Basic LSUN Bedroom model PEEK-G Qualitative Comparison I. Samples are generated by PEEK-G with the google/ddpm-ema-bedroom-256 pixel-space DDPM checkpoint. Best viewed in color.

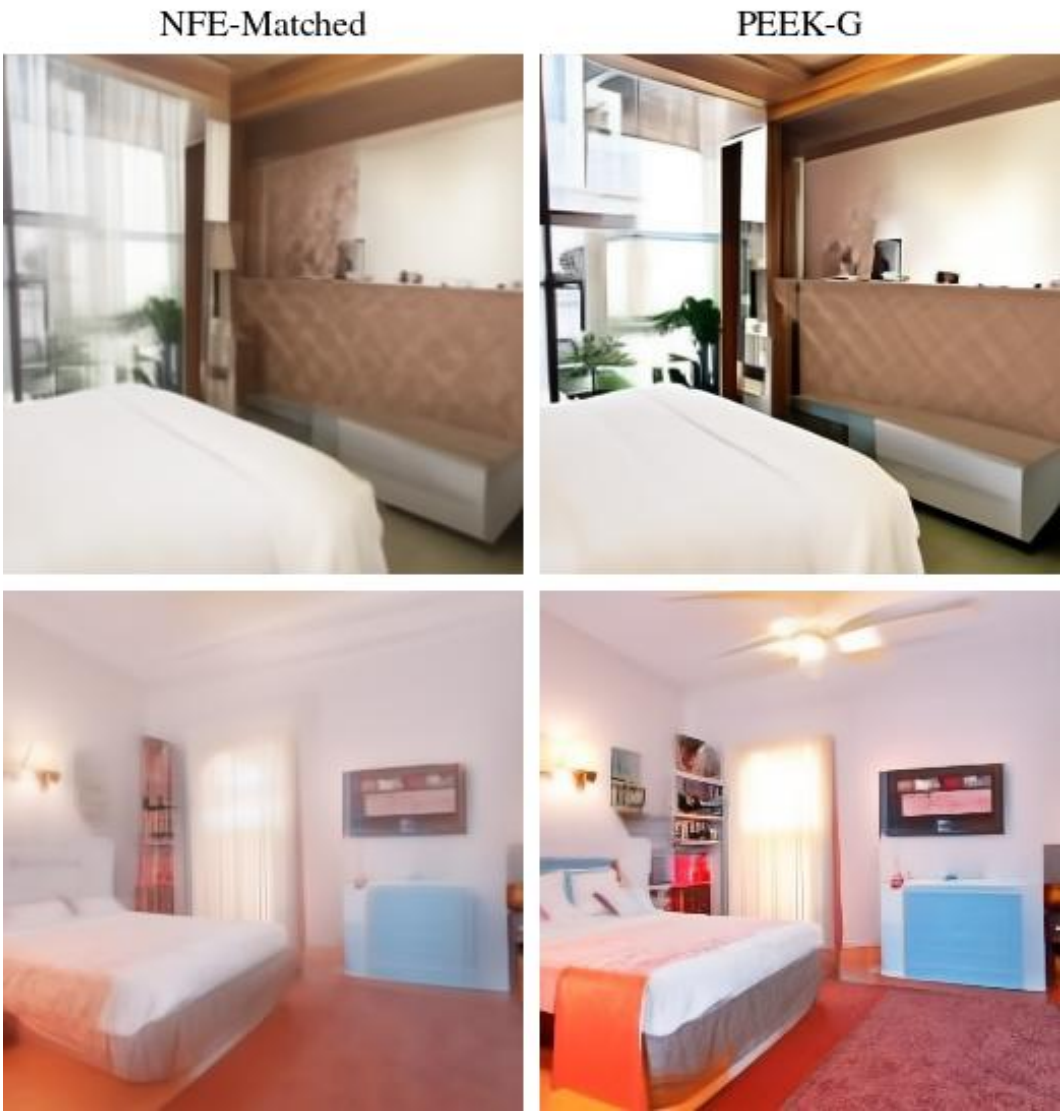

Figure 13: Basic LSUN Bedroom model PEEK-G Qualitative Comparison II. Samples are generated by PEEK-G with the google/ddpm-ema-bedroom-256 pixel-space DDPM checkpoint. Best viewed in color.



Figure 14: ImageNet PEEK-G Qualitative Comparison I. Samples are generated by PEEK-G with the facebook/DiT-XL-2-256 class-conditional latent diffusion transformer with classifier-free guidance. Best viewed in color.



Figure 15: ImageNet PEEK-G Qualitative Comparison II. Samples are generated by PEEK-G with the facebook/DiT-XL-2-256 class-conditional latent diffusion transformer with classifier-free guidance. Best viewed in color.



Figure 16: ImageNet PEEK-G Qualitative Comparison III. Samples are generated by PEEK-G with the facebook/DiT-XL-2-256 class-conditional latent diffusion transformer with classifier-free guidance. Best viewed in color.

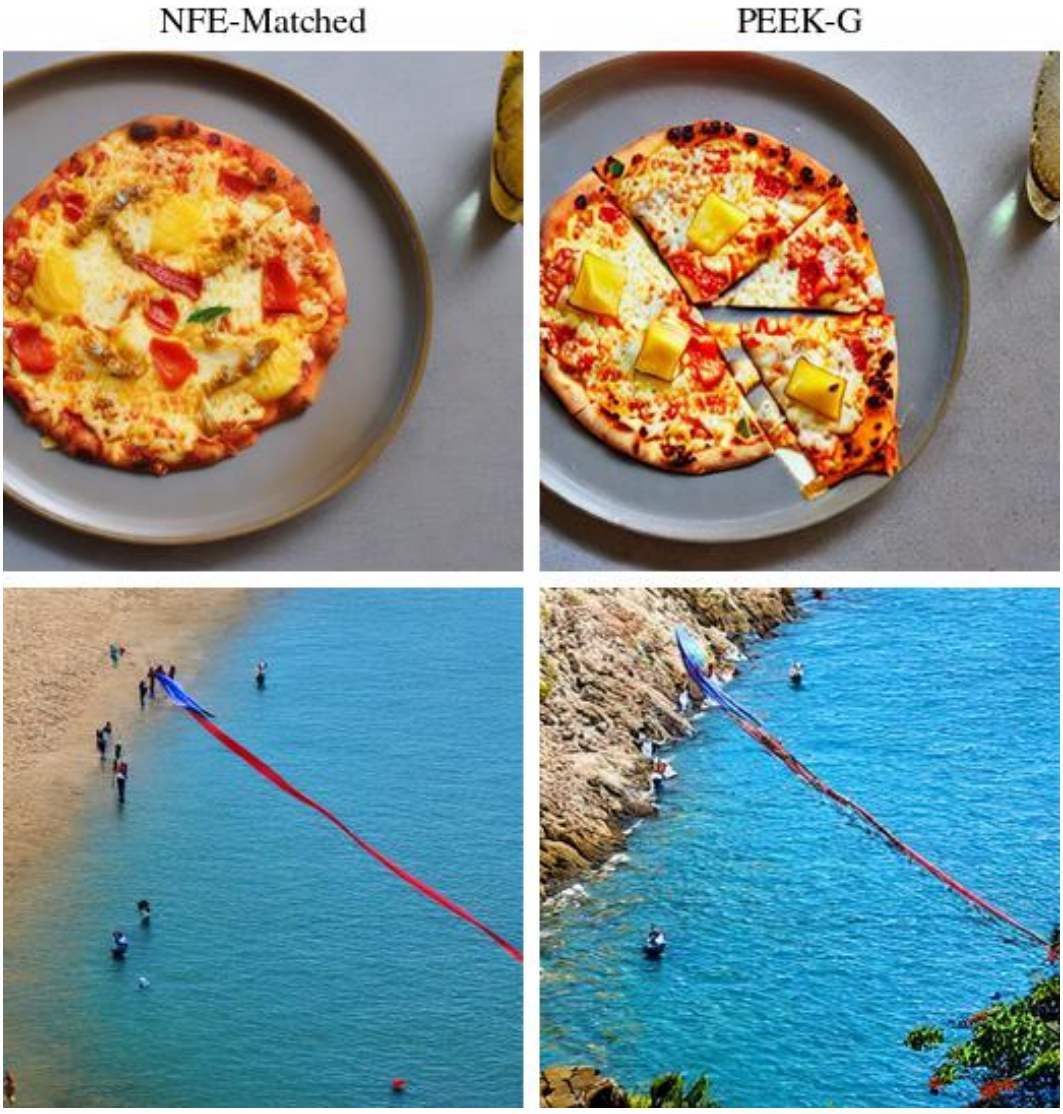

Figure 17: MS-COCO PEEK-G Qualitative Comparison I. Samples are generated by PEEK-G with the Stable Diffusion v1.5 latent text-to-image checkpoint. Best viewed in color.



Figure 18: MS-COCO PEEK-G Qualitative Comparison II. Samples are generated by PEEK-G with the Stable Diffusion v1.5 latent text-to-image checkpoint. Best viewed in color.

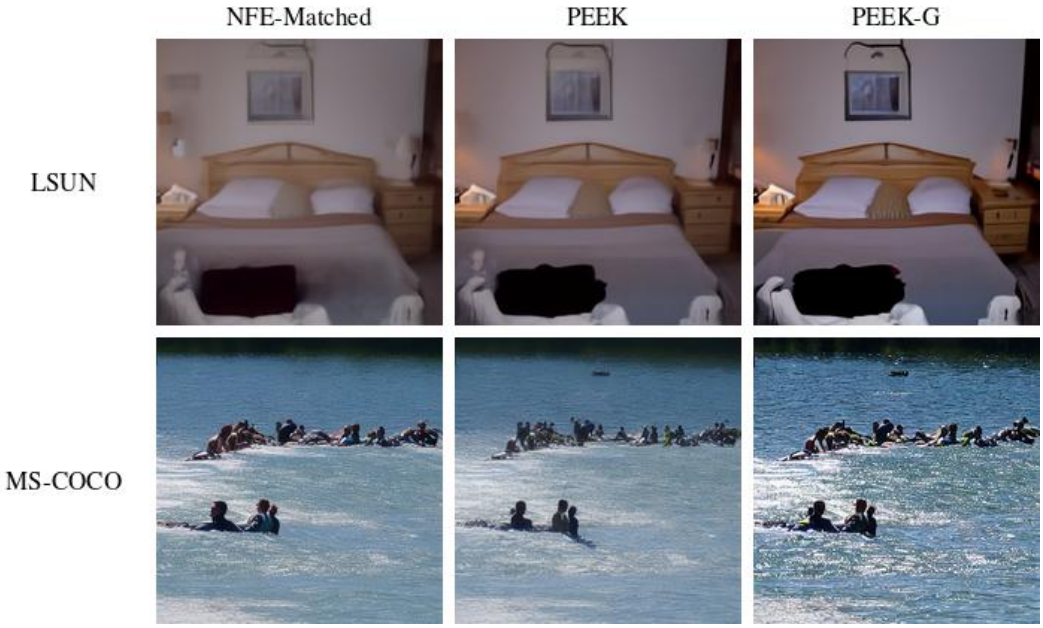

Figure 19: PEEK and PEEK-G Qualitative Contrast. Rows show the basic LSUN Bedroom model and MS-COCO. Columns show the NFE-matched DPM-Solver++ baseline, vanilla PEEK, and PEEK-G. Best viewed in color.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state that PEEK is a training-free deterministic-sampling plugin and limit the empirical claims to the evaluated model, sampler, and dataset settings.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The conclusion reports the observed coverage trade-off, and the experiments and Appendix delimit the evaluated model, sampler, and dataset settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper does not claim a new formal theorem. It gives an implementation-level cache derivation in the main text and Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The method, interpolation schedule search, cache derivation, model families, baselines, and metrics are described in the main paper and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes].

Justification: Code and reproduction instructions are available through the project website cited in the abstract and Appendix G.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The experiments section specifies datasets, model classes, samplers, guidance settings at a high level, metrics, proxy-evaluation settings where used, and compute; further implementation details and selected schedules are provided in the Appendix and through the project website.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

   Justification: The tables report point estimates under matched evaluation settings without error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No].

   Justification: The experiments section reports the GPU class and distributed setup, and the Appendix gives representative calibration time and memory measurements, but per-experiment and total project compute are not fully reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes].

   Justification: The work uses public image-generation benchmarks and pretrained models for an inference-method study, and the Appendix discusses responsible use.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The Appendix notes both positive efficiency/sample-quality implications and possible misuse of improved image generation systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a new high-risk model or dataset; safeguards should follow those of the underlying pretrained generators.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the original sources for the datasets, pretrained models, solvers, and evaluation metrics and does not redistribute datasets or checkpoints, but it does not enumerate every upstream license in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The released implementation is accompanied by reproduction instructions, schedules, commands, and evaluation utilities through the project website.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing or new human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing or new human-subject experiments.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLMs were used only for writing, editing, and formatting assistance, not as an original or non-standard component of the method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.