

AsyncDiff: Asynchronous Timestep Conditioning for Enhanced Text-to-Image Diffusion Inference

Longhuan Xu
Southeast University-Monash University
Joint Graduate School
l.h.xu@outlook.com

Feng Yin*
Southeast University
yinfeng@seu.edu.cn

Cunjian Chen*
Monash University
Cunjian.Chen@monash.edu

Abstract

Text-to-image diffusion inference typically follows synchronized schedules, where the numerical integrator advances the latent state to the same timestep at which the denoiser is conditioned. We propose an asynchronous inference mechanism that decouples these two, allowing the denoiser to be conditioned at a different, learned timestep while keeping image update schedule unchanged. A lightweight timestep prediction module (TPM), trained with Group Relative Policy Optimization (GRPO), selects a more feasible conditioning timestep based on the current state, effectively choosing a desired noise level to control image detail and textural richness. At deployment, a scaling hyper-parameter can be used to interpolate between the original and desynchronized timesteps, enabling conservative or aggressive adjustments. To keep the study computationally affordable, we cap the inference at 15 steps for SD3.5 and 10 steps for Flux. Evaluated on Stable Diffusion 3.5 Medium and Flux.1-dev across MS-COCO 2014 and T2I-CompBench datasets, our method optimizes a composite reward that averages Image Reward, HPSv2, CLIP Score and Pick Score, and shows consistent improvement.

1. Introduction

Diffusion and flow-matching models are now the dominant approach for text-to-image generation [1]. A defining aspect of diffusion models is *timestep conditioning*: the denoiser is queried at a scalar time that indicates current noise level. By default, inference has two parts: the numerical integrator advances the latent along a fixed grid, and the denoiser is queried at the starting time point of each step [15]. We consider it **synchronous inference** because image update and velocity prediction typically follow the same timestep schedule. While this timestep coupling is consistent with the training objective of diffusion models, it seems

mismatched to diffusion dynamics: image latent is updated over a *time interval*, which is the difference between current timestep and next timestep; but velocity prediction is obtained at a single timestep, which assumes that the prediction from diffusion model conditioned at starting noise level is accurate enough to be used throughout the whole update interval. Such assumption can be sub-optimal at inference especially when step interval is large and noise level varies a lot.

A common approach to address this problem is using log-SNR scheduling [6], which aims to make the per-step change in noise level more uniform. Then why not pick a timestep somewhere else within the update interval for velocity prediction? The principal drawback is that it breaks the consistency between training objective and inference in Eqn. (3) because the model is exactly trained to predict current noise of image latent. However, this can be mitigated by keeping image latent at the interval’s start while conditioning the denoiser on a pseudo-timestep chosen to better balance the change in noise level, thus termed as **asynchronous inference**. Furthermore, we argue that the conditioning timestep fed to the denoiser can serve as a post-control signal suggesting the amount of noise retained at each step. Intuitively, conditioning as if the latent were slightly “earlier” or “later” in the schedule changes how the model interprets the latent, thereby modulating textural richness and detail level without altering the base generator. For example, conditioning at a later (cleaner) pseudo-timestep encourages the model to treat more of the current latent as data rather than noise, resulting in extraneous content in final image.

To sum up, asynchronous inference has three notable advantages. First, it reduces the aforementioned conflict: conditioning at a pseudo-timestep inside the update interval better reflects the effective noise seen over that step. Second, it opens a new control axis for image detail independent of base model for trading detail against cleanliness. Third, it improves evaluation scores in practice and reveals the preference of some metrics. Nevertheless, choosing

*Corresponding authors: Cunjian Chen, Feng Yin.

that pseudo-timestep is nontrivial. The evaluation metrics we care about are non-differentiable, and the best conditioning time can be dependent on image latent, predicted velocity, as well as prompt condition. Therefore, a fixed schedule or a single global scale is inadequate. Even when a learned policy suggests nearly constant pseudo-timestep selection, the promising magnitude is unknown ahead of experiments and possibly non-linear in search space. All these challenges motivate a reinforcement learning (RL) approach that optimizes trajectory-level reward by discovering an adaptive policy for asynchronous inference while keeping diffusion model frozen.

In this paper, we introduce a reinforcement learning guided asynchronous diffusion inference method for image generation. A lightweight timestep prediction module (TPM) trained with reinforcement learning selects a pseudo-timestep, not necessarily the starting point of update interval, for velocity prediction conditioning. Velocity prediction schedule and image update schedule are thus naturally de-synchronized. Note that as explained above, we change only the timestep not image latent among the inputs to diffusion model. Our method integrates seamlessly with existing schedulers like DPM-Solver [24] and is controllable at deployment by interpolating between selected pseudo-timestep and reference (original) timestep. We summarize our contributions as follows:

- **Asynchronous diffusion inference:** We are the first to propose the de-synchronization of velocity prediction schedule and image update schedule, which brings consistent improvement on reward metrics.
- **RL-trained timestep predictor:** We train a lightweight timestep prediction module through reinforcement learning, re-parameterizing timestep linearly and locally with a relative, stepwise scaler.
- **Controllable re-timing:** A single interpolation hyperparameter that governs the aggressiveness of de-synchronization and implicitly controls image detail level, sometimes even without RL guidance.
- **Text-to-image metric failure mode exposure:** We reveal a certain high frequency noise pattern which systematically escapes detection by some commonly used evaluation metrics.

2. Related Works

2.1. Flow-matching Models

Rectified flows. Flow-matching models [9, 10] learn the continuous normalizing flow directly, which drives the sample from Gaussian distribution into target data distribution. It achieves state of the art result with typically fewer inference steps than diffusion models [4], and has become the dominant method of recent image generation models.

Inference scheduling. Classical diffusion samplers such as DDPM [4] and DDIM [14] traverse a fixed grid of T discrete noise levels, i.e., $t = 0, 1, \dots, T$, corresponding to a linear ramp of the variance parameters β_t . Flow matching models typically adopt a grid that is uniform in log signal-noise ratio (SNR) space, an empirically robust compromise that balances inference speed and image quality. Recent works [19, 21, 22] have explored data-adaptive schedules, yet these methods keep the velocity prediction and image update schedule synchronized: they modify the grid but not the coupling. Consequently, previous works failed to explore the complicated de-synchronized schedule space where diffusion condition timestep can diverge away from image latent timestep.

2.2. Text-to-Image Evaluation

Evaluating text-to-image generation involves two complementary goals: (i) *text to image alignment*: how faithfully an image reflects the semantics and style of the prompt, and (ii) *prompt-agnostic perceptual quality*. A variety of neural network based metrics have been developed. CLIP score [11] was trained with large scale contrastive learning; HPS [17], ImageReward [18] and PickScore [7] are explicitly tuned on extensive human-preference data and feedback.

However, our experiments suggest that some existing metrics are still imperfect: they tend to reward increased local detail and structure but can be relatively insensitive to subtle high-frequency artifacts or low-amplitude Gaussian background noise.

2.3. Reinforcement Learning

RL has emerged as a powerful tool for steering large generative models toward non-differentiable goals such as human preference, safety, or sampling efficiency. For text-to-image generation, the sampler can be cast as an *agent* that takes action (in our case choosing the next timestep) given current state including image latent, predicted velocity, step index and prompt embedding, and receives a scalar reward once a clean image is rendered.

PPO. Proximal Policy Optimization (PPO) [12] is one of the workhorse algorithm in reinforcement learning. It stabilizes training and improves final performance by simply clipping the original objective.

3. Proposed Method

In this section, we first briefly go through how images are normally sampled from flow matching T2I diffusion models. Second, we explain the core idea behind our RL-trained sampler: timestep de-synchronization of image update and velocity prediction. Finally, we provide details of the implementation.

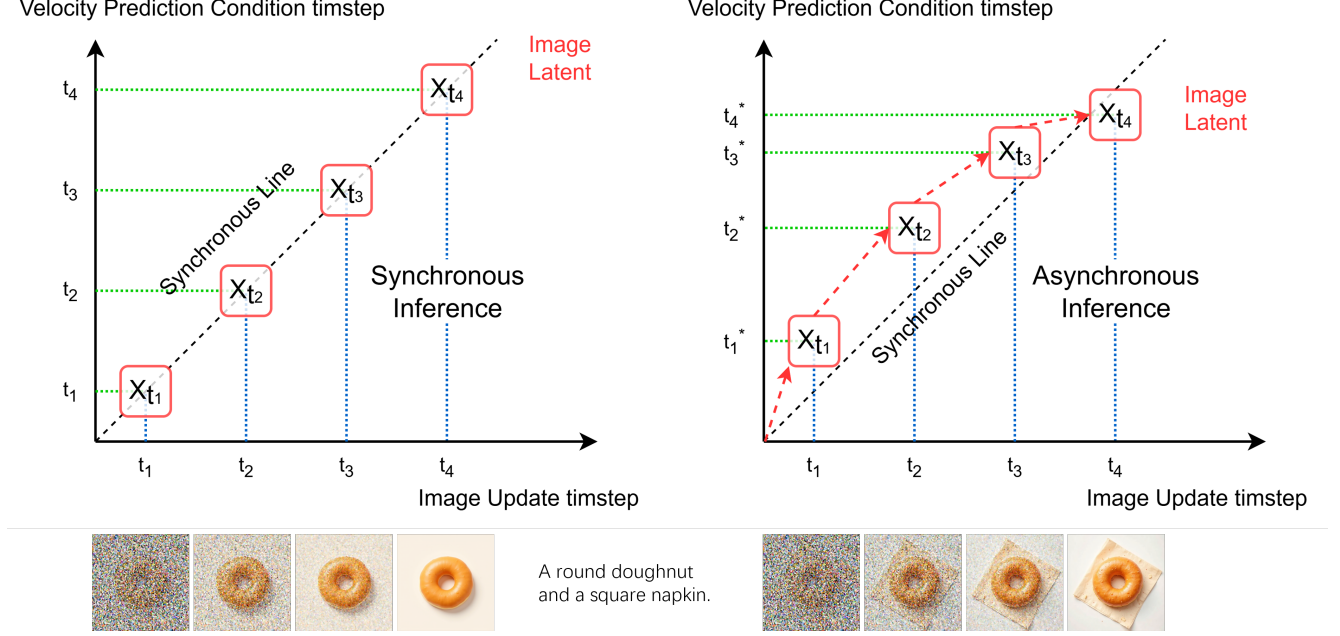


Figure 1. Asynchronous Inference.

Algorithm 1 Default flow-matching sampler

Input: prompt p , steps K , schedule \mathcal{T} , guidance scale ω

Output: image y

```

1:  $\mathbf{c} \leftarrow \text{TextEncoder}(p)$ 
2:  $\mathbf{x}_{t_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
3: for  $k = 0$  to  $K - 1$  do
4:    $\mathbf{v}_{\text{cond}} \leftarrow f_{\theta}(\mathbf{x}_{t_k}, t_k, \mathbf{c})$ 
5:    $\mathbf{v}_{\text{uncond}} \leftarrow f_{\theta}(\mathbf{x}_{t_k}, t_k, \mathbf{0})$ 
6:    $\mathbf{v}_k \leftarrow \mathbf{v}_{\text{uncond}} + \omega(\mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{uncond}})$ 
7:    $\mathbf{x}_{t_{k+1}} \leftarrow \mathbf{x}_{t_k} + (t_{k+1} - t_k) \mathbf{v}_k$ 
8: end for
9:  $\mathbf{y} \leftarrow \text{VAEDecoder}(\mathbf{x}_{t_K})$ 
10: return  $\mathbf{y}$ 

```

3.1. Preliminary

Diffusion inference for image generation. Classic diffusion models define a forward process that gradually adds Gaussian noise to data with a predefined time-dependent variance schedule β_t [4, 15]. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (1)$$

Flow-matching/rectified-flow models directly learn the velocity field v_t defined as the time derivative of image latent¹ and integrate a deterministic ODE [9] at inference based on

¹We work in the VAE latent space; the final RGB image is recovered with the pretrained decoder.

a monotonically-decreasing time grid $\mathcal{T} = \{t_0 = 1 > t_1 > \dots > t_K = 0\}$:

$$v_t = \frac{d\mathbf{x}_t}{dt} = f_{\theta}(\mathbf{x}_t, t, \mathbf{c}), \quad t = t_k, \quad k = 0, 1, \dots, K. \quad (2)$$

where f_{θ} predicts the *velocity* (or instantaneous drift) conditioned on current image latent \mathbf{x}_t , timestep t_k and text embedding \mathbf{c} . Denoting ϵ as gaussian noise, velocity training follow flow-matching loss:

$$\mathcal{L} = \mathbb{E} \left[\left\| \mathbf{v}_t(\mathbf{x}_t, t, \mathbf{c}) - (\epsilon - \mathbf{x}_0) \right\|_2^2 \right] \quad (3)$$

With an explicit Euler discretization, during inference one image update step reads:

$$\mathbf{x}_{t_{k+1}} = \mathbf{x}_{t_k} + \Delta t_k f_{\theta}(\mathbf{x}_{t_k}, t_k, \mathbf{c}), \quad \Delta t_k = t_{k+1} - t_k. \quad (4)$$

The same timestep t_k is used both for velocity prediction and image update which is why we call this classic method **synchronous inference**.

Classifier-free guidance. For better text-image alignment, the sampler may employ classifier-free guidance (CFG) [3]. Two velocity predictions are performed per step, f_{θ}^{cond} using text condition \mathbf{c} and $f_{\theta}^{\text{uncond}}$ with the null embedding, and are linearly combined as:

$$v_{\theta}^{\text{CFG}} = v_{\theta}^{\text{uncond}} + \omega(v_{\theta}^{\text{cond}} - v_{\theta}^{\text{uncond}}). \quad (5)$$

Here, v_{θ}^{cond} and $v_{\theta}^{\text{uncond}}$ denote velocity predictions with and without text conditioning, v_{θ}^{CFG} is the guided velocity actually used in image update, and ω is the guidance scale.

Algorithm 2 Asynchronous Sampler (TPM)

Input: prompt p , max steps K_{\max} , σ_{\min} , guidance scale ω **Output:** image y

```
1:  $\mathbf{c} \leftarrow \text{TextEncoder}(p)$ 
2: Initialize  $\mathbf{x}_{t_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $t_0^* = t_0 = 1$ 
3: for  $k = 0$  to  $K_{\max} - 1$  do
4:    $\mathbf{v}_{\text{cond}} \leftarrow f_{\theta}(\mathbf{x}_{t_k}, t_k^*, \mathbf{c})$ 
5:    $\mathbf{v}_{\text{uncond}} \leftarrow f_{\theta}(\mathbf{x}_{t_k}, t_k^*, \mathbf{0})$ 
6:    $\mathbf{v}_{\mathbf{k}}^* \leftarrow \mathbf{v}_{\text{uncond}} + \omega(\mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{uncond}})$ 
7:    $\mathbf{x}_0^k \leftarrow \mathbf{x}_k - t_k \mathbf{v}_{\mathbf{k}}^*$ 
8:    $(\alpha_{k+1}^*, \beta_{k+1}^*) \leftarrow \text{TPM}_{\theta}(\mathbf{x}_{t_k}, \mathbf{v}_{\mathbf{k}}^*, t_k^*, \mathbf{x}_0^k, \mathbf{c}, k)$ 
9:    $r_k^* \sim \text{Beta}(\alpha_k^*, \beta_k^*)$ 
10:   $t_{k+1}^* \leftarrow \max(\sigma_{\min}, t_k + (0.5 + r_k^*)(t_{k+1} - t_k))$ 
11:   $\mathbf{x}_{t_{k+1}} \leftarrow \mathbf{x}_{t_k} + (t_{k+1} - t_k) \mathbf{v}_{\mathbf{k}}^*$ 
12: end for
13:  $\mathbf{x} \leftarrow \mathbf{x}_{t_{k+1}} - t_{k+1} \mathbf{v}_{k+1}^*$ 
14:  $y \leftarrow \text{VAEDecoder}(\mathbf{x})$ 
15: return  $y$ 
```

Timestep selection. In diffusion (and flow) samplers, timestep selection means choosing a timestep grid at which the velocity field is evaluated to numerically simulate the reversed sampling process. It is a discretization schedule of a fixed number of function evaluations (NFE). In applications where fewer steps are favored, the choice of timestep can become the dominant factor for stability and fidelity at inference time. Classical inference timestep schedules include model-agnostic uniform linear schedule and log-SNR schedule, typically paired with high-order samplers such as DPM-Solver [24] and UniPC [23]. Recent works not only optimize the grid conditioned on the pretrained model [16] but also make it online-adaptive to the data [22].

Our asynchronous inference likewise selects timestep conditioned on both diffusion model and input data using a timestep prediction module (TPM) which iteratively outputs next timestep prediction.

3.2. Asynchronous Inference

Decoupling image update and velocity prediction. To relax the coupling between image update schedule and velocity prediction schedule in Eqn. (2) and Eqn. (4), we let the sampler query velocity field conditioned at a *velocity-prediction timestep* t_k^* that can differ from *latent-update timestep* t_k which is used to advance the sample. Since it is infeasible to enumerate all possible timestep combinations, our goal is to find a general de-synchronized scheduling policy during sampling to yield high quality images. Concretely, learnable velocity prediction pseudo-timestep t_k^* is parameterized as:

$$t_k^* = t_{k-1} + (0.5 + r_k^*) \Delta t_{k-1}, \quad \Delta t_{k-1} = t_k - t_{k-1}, \quad (6)$$

which not only takes original schedule as reference baseline but also has linear and stepwise deviation scaling. The multiplicative ratio r_k^* is drawn from beta distribution, $r_k^* \sim \text{Beta}(\alpha_k^*, \beta_k^*)$, defined by the output of timestep prediction module (TPM):

$$(\alpha^*, \beta^*) = \text{TPM}_{\theta}(x_{k-1}, v_{k-1}^*, t_{k-1}^*, x_0^{k-1}, \mathbf{c}, k). \quad (7)$$

As described in Eqn. (7), TPM takes several inputs: previous latent image \mathbf{x}_{k-1} , previous velocity \mathbf{v}_{k-1}^* , text (prompt) condition \mathbf{c} , calculated clean image \mathbf{x}_0^{k-1} , previous pseudo-timestep t_{k-1}^* and step index k indicating position along the whole inference process².

Remember the trick of asynchronous inference is to keep image latent untouched while presenting the denoiser with a dynamically chosen pseudo-timestep t_k^* , which implicitly sets the desired noise level and naturally desynchronizes the process. Now modifying only the timestep term in (2) gives the de-synchronized velocity:

$$v_k^* = f_{\theta}(\mathbf{x}_{t_k}, t_k^*, \mathbf{c}) \quad (8)$$

Accordingly, Eqn. (4) becomes:

$$\mathbf{x}_{t_{k+1}} = \mathbf{x}_{t_k} + \Delta t_k v_k^*, \quad \Delta t_k = t_{k+1} - t_k \quad (9)$$

Here t_k and t_{k+1} denote untouched image update schedule. In Eqn. (6), when $r_k^* = 0.5$, asynchronous inference falls back to synchronous inference, and asynchronous pseudo-timestep value range relative to synchronous timestep is given by:

$$\eta = 0.5 + r_k^* \quad (10)$$

Consequently, we define asynchronous deviation as:

$$D = r_k^* - 0.5 \quad (11)$$

To enable controllable de-synchronization at deployment, we introduce the scaling of deviation:

$$\begin{aligned} \eta_{\text{scaled}} &= 1 + (r_k^* - 0.5) \times \gamma \\ D_{\text{scaled}} &= (r_k^* - 0.5) \times \gamma \end{aligned} \quad (12)$$

where γ is the scaling hyper-parameter. Specifically, We conducted a comparative experiment where the manual bound on deviation is amplified into $[-1, 1]$ by:

$$\eta' = 2 \times r_k^*, \quad D' = 2 \times (r_k^* - 0.5) \quad (13)$$

This comparative experiment is different from post-scaling Eqn. (12) as it involves training TPMs again under new constraint.

²Although clean image can be computed mathematically, passing it in directly relieves the network from having to reconstruct this exact transformation.

Table 1. SD3.5-Medium (512×512), half precision (FP16), 15 steps.

Dataset	Method	ImageReward	HPSv2	CLIP	PickScore	Mean Deviation
MS-COCO 2014	SD Base	0.9043	0.2754	<u>0.2683</u>	22.27	0
	Our(0.5 + r^*)	<u>0.9243</u>	<u>0.2766</u>	0.2685	22.28	+0.192
	Our(2 × r^*)	0.9670	0.2798	0.2669	22.28	+0.398
T2I-CompBench	SD Base	0.8642	0.2698	0.2754	22.09	0
	Our(0.5 + r^*)	<u>0.8969</u>	<u>0.2722</u>	0.2754	22.11	+0.231
	Our(2 × r^*)	0.9377	0.2729	0.2757	<u>22.10</u>	+0.106

Table 2. Flux.1-dev (512×512), half precision (FP16), 10 steps.

Dataset	Method	ImageReward	HPSv2	CLIP	PickScore	Mean Deviation
MS-COCO 2014	Flux Base	0.7791	0.2769	0.2592	22.60	0
	Ours(0.5 + r^*)	<u>0.9267</u>	<u>0.2871</u>	<u>0.2600</u>	22.74	+0.499
	Ours(2 × r^*)	0.9452	0.2920	0.2601	<u>22.70</u>	+0.959
T2I-CompBench	Flux Base	0.7197	0.2747	0.2655	22.46	0
	Ours(0.5 + r^*)	0.8725	<u>0.2854</u>	0.2673	22.57	+0.499
	Ours(2 × r^*)	<u>0.8584</u>	0.2866	<u>0.2667</u>	<u>22.52</u>	+0.640

Trajectory optimization. We train TPM with *Group-Relative PPO* (GRPO) [13] which, for each prompt, generates multiple samples and computes normalized advantages using the group mean and variance. Compared with PPO:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E} \left[\min(r \hat{A}, \text{clip}(r, 1-\varepsilon, 1+\varepsilon) \hat{A}) \right], \quad r = \frac{\pi_{\theta}(\tau)}{\pi_{\text{old}}(\tau)}. \quad (14)$$

GRPO removes the need for an explicit value network while preserving the variance-reduction benefits of a baseline. During training, we roll out a *full* sampling loop until termination that alternates between frozen diffusion model and trainable TPM. The resulting schedule $\hat{\mathcal{T}} = (t_1, \dots, t_N)$ is a complete trajectory whose log-probability factorizes as:

$$\pi_{\theta}(\hat{\mathcal{T}}) = \prod_{i=1}^N \pi_{\theta}^{(i)}(t_i) \quad (15)$$

The whole trajectory is treated as a single action, on which we do PPO-clip with a trajectory-level surrogate. It terminates when (i) a global step cap $K_{\max} = 15$ (SD3.5-medium) or 10 (Flux.1-dev) is reached, or (ii) $t_{k+1} < \sigma_{\min} = 10^{-3}$ for all samples in the batch. A final Euler step brings the latent to $t = 0$.

Composite RL objective. Each trajectory is assigned a scalar reward that averages over four diverse metrics: Image Reward, HPSv2, CLIP Score and Pick Score. Each metric is z-score normalized within the batch to ensure numerical balance.

$$R = \frac{1}{4} \sum_{i \in \{\text{IR}, \text{HPS}, \text{CLIP}, \text{Pick}\}} \hat{S}_i, \quad \hat{S}_i = \frac{\text{Score}_i - \mu_i}{\sigma_i + \varepsilon} \quad (16)$$

Timestep Prediction Module architecture. TPM adopts a *token-centric* design that casts model input into a short transformer sequence. Drop-out and layer-normalization are disabled to avoid unnecessary randomness or complexity and smooth training.

- Latent tokens.** The three spatial tensors—noisy latent, predicted flow and clean image—are first cut into patches and then projected into a joint embedding space.
- Condition tokens.** A handful of condition tokens are appended to the sequence, including *Text-condition tokens* obtained by pre-processing the prompt and *temporal tokens* that encode image update timestep.
- Backbone.** All tokens are fed to a shallow 4-layer transformer encoder.
- Read-out head.** The final set of global tokens is flattened and passed through a small MLP that emits two scalars (a^*, b^*). Each scalar is mapped to a strictly positive value via

$$\phi(x) = \begin{cases} 2 + x + \frac{1}{2}x^2, & x > 0, \\ 1 + e^x, & x \leq 0, \end{cases} \quad (17)$$

producing parameters (α^*, β^*) required in Eqn. (7) and ensuring a uni-modal Beta distribution.

4. Experimental Results

4.1. Experimental Setup

Implementation details. We use publicly available Flux.1-dev and SD 3.5-medium text-to-image flow matching models in **half-precision** (FP16), generating 512 × 512 images. Gradient is normalized at 1.0 and classifier-free guidance scale is set to 5. Inference is capped at $K_{\max} = 10$

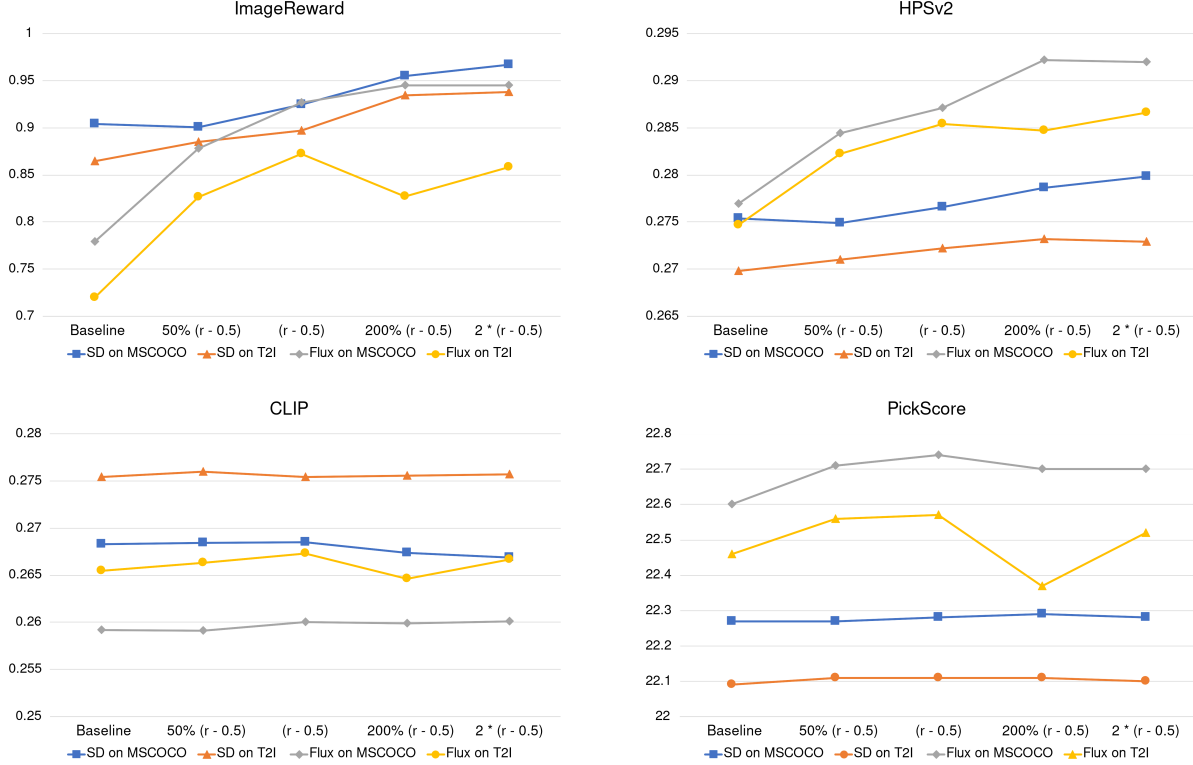


Figure 2. Deviation scaling result. Each chart plots evaluation metrics as we scale the per-step deviation ($r_k^* - 0.5$) from (11) using the factor γ in (12) by 50%, 100% and 200%. Synchronous inference baseline and comparative experiments with lifted deviation bound (13) denoted as $2 * (r_k^* - 0.5)$ are also covered. Curves include ImageReward, HPSv2, CLIP, and PickScore (all higher is better) on MS-COCO 2014 and T2I-CompBench from SD and Flux models.

steps (Flux) or 15 steps (SD 3.5). We adopt PPO clip ratio $\varepsilon = 0.2$ and constant learning rates: $2e-5$ for Flux and a smaller $1e-5$ for SD since SD undergoes more inference steps and we do trajectory-level optimization.

Datasets and training. Prompts are drawn from the MS-COCO 2014 captions [8] and T2I-CompBench dataset [5], following official training-validation split. Specifically, MS-COCO 2014 provides over 400k training captions and over 200k validation captions, while T2I-CompBench offers $\sim 5.5k$ training prompts and $\sim 2.4k$ validation prompts. We fix the random seed to 42, randomly sample 2,048 MS-COCO validation prompts, and use the entire T2I-CompBench validation set. Training uses the full training splits of both datasets. In each iteration, we draw a single prompt and pair it with 16 Gaussian noise latents, then apply GRPO with a mini-batch size of 4, using the mean score over the 16 rendered images as the baseline to reduce inter-prompt variance and stabilize the advantage estimates. For both datasets and for both SD and Flux backbones, the TPM is trained for 4k iterations (i.e., 4k prompts \times 16 image samples).

Evaluation metrics. We report four automatic metrics used both for training and evaluation: (i) ImageReward [18], a reward model distilled from human feedback that correlates with human judgments of aesthetic quality and alignment; (ii) HPSv2 [17], a learned preference model trained on large-scale human comparisons that reflects perceived image quality and prompt faithfulness; (iii) CLIP Score [11], the cosine similarity between CLIP image and text embeddings, which primarily measures semantic alignment between the prompt and the generated image; and (iv) PickScore [7], a preference predictor trained on the Pick-a-Pic dataset capturing overall human-perceived appeal.

4.2. Main Results

During testing or validation, r_k^* is no longer a random variable, but instead a deterministic value given by:

$$r_k^* = \arg \max_{r \in (0,1)} r^{\alpha_k^* - 1} (1 - r)^{\beta_k^* - 1} = \frac{\alpha_k^* - 1}{\alpha_k^* + \beta_k^* - 2}, \quad (18)$$

which is well-defined here since (17) ensures $\alpha_k^* > 1$ and $\beta_k^* > 1$. The value of r_k^* ranges over $[0, 1]$, thus deviation is constrained within $[-0.5, 0.5]$. Empirically, we notice that on Flux models the TPM frequently indeed hits the $+0.5$ de-

violation ceiling when deviation bound is not lifted (11), indicating that Flux is more adaptable to asynchronous timestep drift, and that our comparative experiments (13) are necessary.

Main results for synchronous inference baseline and asynchronous inference with or without lifted bound are reported in Table 1 and Table 2. Quantitatively, compared against default synchronized sampling, our asynchronous inference yields consistent metric gains across datasets and models. All TPMs tend to push denoise timestep forward (closer to clean image time point) as implied by positive deviation value, effectively presenting a lower noise level to the denoiser for extra detail/content. This agrees with our initial guess that choosing a denoising timestep somewhere in the middle of image update interval may be beneficial.

Nevertheless, Flux benefits much more from asynchronous inference, likely for two reasons: (i) the Flux model is more robust and better at handling excessive noise; and (ii) it uses fewer inference steps, yielding larger timestep intervals and therefore a more serious misalignment between velocity-prediction time point and image-update interval. In other words, **asynchronous inference may yield great metric gains when the total number of inference steps is small with powerful diffusion backbone models.**

We also observe clear trade-offs among the four metrics: improving one can depress another. Xue et al. [20] report a similar tension—training Flux solely on HPSv2 reduces CLIP and GenEval [2]. In our case, asynchronous inference with position deviation tends to increase ImageReward and HPSv2 while slightly reducing CLIP and PickScore. In comparative experiments, lifting the bound of deviation indeed achieves better ImageReward and HPSv2, but is at the risk of suffering a severe drop in CLIP and PickScore.

Figs (3) (4) show the metric results of scaling the original deviation capped within $[-0.5, 0.5]$. Together with qualitative visualization in Figs (5), **we argue that larger (positive) deviations yield richer, more detailed—and sometimes busier—images favored by ImageReward and HPSv2, whereas CLIP and PickScore do not always benefit, because some unsolved Gaussian background noise would also be left in the image.** This underscores that different metrics emphasize different aspects of quality, and that **many metrics failed to take high frequency noisy patterns in final image into consideration** but are instead satisfied with low level details, while other metrics focus on semantic prompt alignment.

From left to right within each panel in Figs (5), the images correspond to: baseline (synchronized), 50% scaled, 100% scaled, 200% scaled version of $(r - 0.5)$ deviation (12), and comparative $2 \times (r - 0.5)$ deviation (13). In some cases like the first, second and third rows in Figs (5), the denoiser is allowed the chance to denoise missing objects and

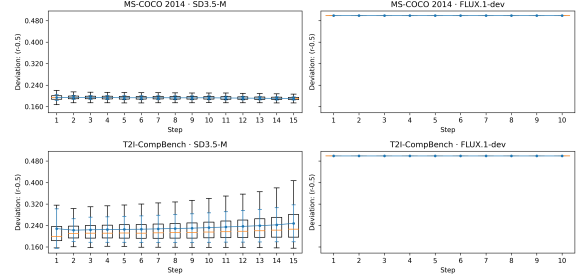


Figure 3. Per-step deviation: $(r - 0.5)$ distributions across datasets/models.

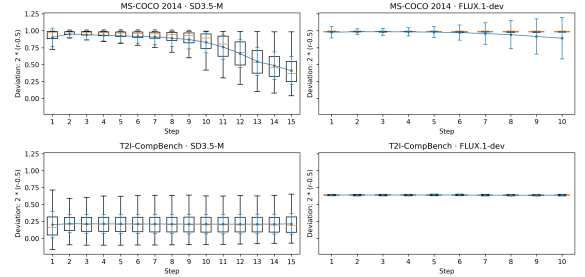


Figure 4. Per-step deviation of comparative experiments: $2 \times (r - 0.5)$.

fix errors; in general image detail level and textural richness are increased, with the annoying drawback of leaving some unsolved Gaussian noise in background, as shown in the fourth, fifth and sixth rows in Figs (5). Although the metric gains can be occasionally modest, the qualitative improvement of crisper textures and richer structure is consistent and clearly visible.

Besides, the deviation of each step varies along the trajectory, and TPMs trained on different datasets or backbones adopt distinct policies Figs (3, 4): Flux shows near-constant deviation across steps with low variance, whereas SD exhibits higher variance with sometimes a characteristic pattern—large early deviations that gradually taper toward later steps. **When the per-step deviation exhibits low variance, an RL-free policy with a fixed deviation level—agnostic to sample context—may suffice and simplify deployment.**

5. Alternative Approach and Future Work

The idea of deliberately forcing the diffusion model to treat part of the noise as image data (and vice versa) can also be leveraged in another way. Instead of modifying the inference timestep schedule, an independent scalar ω can be multiplied into Eqn. (4), yielding:

$$\mathbf{x}_{t_{k+1}} = \mathbf{x}_{t_k} + \Delta t_k f_{\theta}(\mathbf{x}_{t_k}, t_k, \mathbf{c}) \omega \quad (19)$$



Figure 5. Qualitative results. Each image in each panel from left to right correspond to: synchronous inference (baseline); 50%, 100% and 200% scaled asynchronous inference with deviation ($r_k^* - 0.5$); comparative asynchronous inference with deviation $2 * (r_k^* - 0.5)$.

Similarly, ω is set within $[0.5, 1.5]$ by:

$$\omega = 0.5 + r_k^*, \quad r_k^* \in [0, 1] \quad (20)$$

Likewise, when ω is smaller than 1, some noise will be left to be interpreted as image data while keeping timestep untouched. Although this alternative approach appears promising, early experiments indicate that it is less stable and more prone to meaningless high-frequency patterns than asynchronous inference, as is qualitatively illustrated in Fig (6).

6. Conclusion

In this paper, we introduced RL-Guided Asynchronous Diffusion Inference, a sampling method that de-synchronizes the denoiser’s conditioning timestep from the latent-update schedule. The approach is lightweight and plug-and-play,



Figure 6. Qualitative comparison of proposed asynchronous inference and alternative approach.

offers a single scaling hyper-parameter for controllable de-synchronization, remains compatible with standard schedulers, and delivers consistent gains across backbones and datasets—particularly on ImageReward and HPSv2—by letting the TPM present a cleaner effective noise level to the denoiser.

There are, however, limitations: RL training is not fully stable—especially for the SD model on MS-COCO—where the learned policy can diverge between lifted and unlifted deviation-bound settings and drift over training. In addition, current reward signals insufficiently penalize residual high-frequency artifacts; a more robust metric that explicitly distinguishes structured detail from spurious noise is needed as an RL objective.

In the future, we will integrate de-synchronization with a wider range of schedulers and high-order solvers, scale training with diverse inference steps and broader datasets to better map quality–efficiency trade-offs, refine the alternative independent scalar by narrowing its value range to suppress unwanted noisy patterns, and combine this multiplicative control with asynchronous timestep conditioning to exploit their complementary strengths.

References

- [1] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yan-nik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. ¹
- [2] Dhruva Ghosh, Hanna Hajishirzi, and Ludwig Schmidt.

- Geneval: An object-focused framework for evaluating text-to-image alignment, 2023. 7
- [3] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. 3
- [4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2, 3
- [5] Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation, 2025. 6
- [6] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2023. 1
- [7] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation, 2023. 2, 6
- [8] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 6
- [9] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. 2, 3
- [10] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. 2
- [11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2, 6
- [12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 2
- [13] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. 5
- [14] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 2
- [15] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. 1, 3
- [16] Vinh Tong, Hoang Trung-Dung, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion odes, 2025. 4
- [17] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023. 2, 6
- [18] Jiazhen Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023. 2, 6
- [19] Shuchen Xue, Zhaoqiang Liu, Fei Chen, Shifeng Zhang, Tianyang Hu, Enze Xie, and Zhenguo Li. Accelerating diffusion sampling with optimized time steps, 2024. 2
- [20] Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. Dancegrpo: Unleashing grpo on visual generation, 2025. 7
- [21] Hancheng Ye, Jiakang Yuan, Renqiu Xia, Xiangchao Yan, Tao Chen, Junchi Yan, Botian Shi, and Bo Zhang. Training-free adaptive diffusion with bounded difference approximation strategy, 2024. 2
- [22] Zilyu Ye, Zhiyang Chen, Tiancheng Li, Zemin Huang, Weijian Luo, and Guo-Jun Qi. Schedule on the fly: Diffusion time prediction for faster and better image generation, 2025. 2, 4
- [23] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models, 2023. 4
- [24] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics, 2023. 2, 4