RA-CFGCache: From Branch-Level Criteria to Guided-Risk Control under Classifier-Free Guidance

Yiming Liu[1,3] **Ben Wan**[2] **Tongxuan Liu**[2] **Ao Wang**[1,3] **Yuqi Xiong**[3]
Fan Zhang[1,3] **Hui Chen**[3,†] **Guiguang Ding**[1,3,†]
[1]School of Software, Tsinghua University, Beijing, China
[2]JD.com, Beijing, China
[3]BNRist, Tsinghua University, Beijing, China

Abstract

Diffusion models enable high-quality visual generation, but iterative denoising remains computationally expensive, especially under classifier-free guidance (CFG), which requires both conditional and unconditional evaluations. Training-free caching reduces this cost by reusing previously computed features or predictions. However, branch-local reuse criteria do not explicitly account for how cache errors combine under CFG or how local perturbations affect the final output. We examine two misalignments in cache control: a *branch–guided mismatch*, where guided error depends on both the magnitudes and alignment of branch errors, and a *local–final mismatch*, where the downstream impact of a local error varies across timesteps. We propose **RA-CFGCache**, a **R**isk-**A**ligned **Caching** framework under **CFG** that incorporates both factors while keeping the sampling schedule and guidance rule fixed. *CFG-aware Guided-Risk Composition* combines existing branch-wise proxies using CFG coefficients and offline-calibrated cross-branch alignment. *Propagation-Aware Rescaling* further weights the resulting guided-risk estimate with a timestep-dependent propagation prior calibrated from isolated reuse perturbations. An online threshold controller then determines when to jointly refresh or reuse both branches. Experiments on FLUX.1-dev, Wan2.1-T2V-1.3B, and CogVideoX-2B demonstrate improved efficiency–fidelity trade-offs over the evaluated training-free caching baselines. Moreover, **RA-CFGCache** is compatible with diverse base proxy families, including TeaCache-, DiCache-, and MagCache-style estimators, and consistently improves fidelity at nearly unchanged latency. Code is available at `https://github.com/yiming-l21/RA-CFGCache.git`.

1 Introduction

Recent years have witnessed substantial progress in diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song and Ermon, 2020, Dhariwal and Nichol, 2021, Peebles and Xie, 2023] for visual generation. In modern conditional diffusion inference, classifier-free guidance (CFG) [Ho and Salimans, 2022] has become a standard mechanism for improving conditional fidelity and generation quality. However, diffusion inference remains computationally expensive because it requires repeated network evaluations over many denoising steps, and explicit two-branch CFG further increases the per-step cost by evaluating both conditional and unconditional predictions.

To reduce this cost, prior work has explored various diffusion acceleration strategies, including distillation [Meng et al., 2023, Sauer et al., 2023], efficient samplers [Song et al., 2022, Lu et al., 2022], pruning [Zhang et al., 2024], and feature caching [Ma et al., 2023, Liu et al., 2025a]. Among them, training-free caching is particularly attractive, as it improves efficiency without retraining or

[†]Corresponding authors.

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

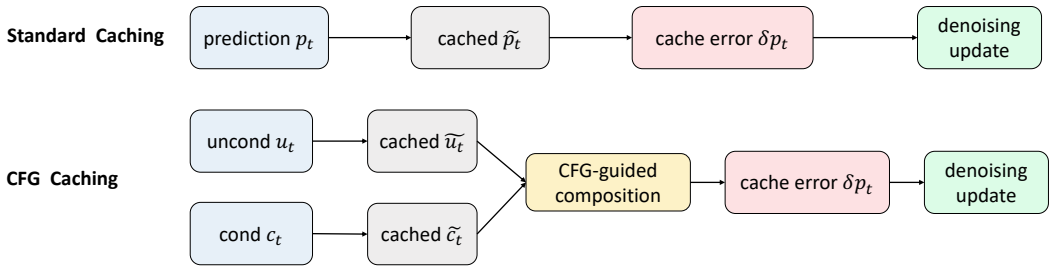
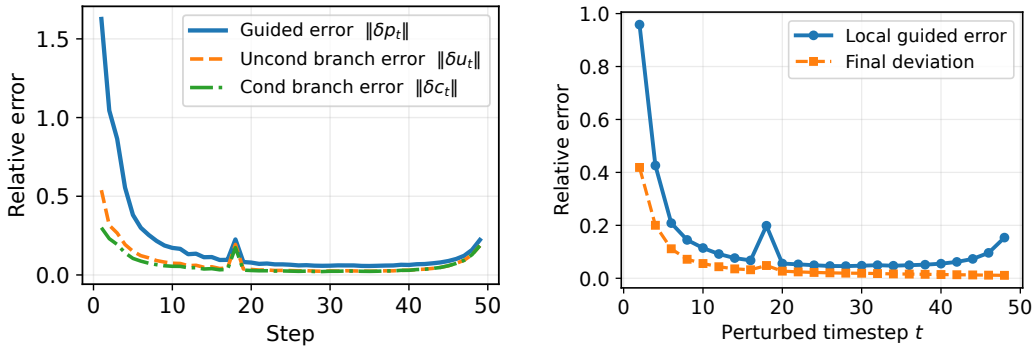


Figure 1: **Standard caching vs. CFG caching.** Standard caching reuses a single prediction whose cache error directly enters the denoising update, whereas CFG caching reuses conditional and unconditional branches whose errors are composed into the guided prediction.



(a) Branch-wise error vs. guided error.

(b) Local guided error vs. final deviation.

Figure 2: **Two mismatches under CFG caching.** (a) The first mismatch: branch-wise cache errors do not directly reflect the guided error that enters the denoising update. (b) The second mismatch: local guided error does not directly predict the final deviation after error propagation across timesteps.

modifying the original model. Since model outputs at nearby timesteps often exhibit substantial redundancy, previously computed features or predictions can be reused when an estimated change remains sufficiently small [Ma et al., 2023, Liu et al., 2025a, Ma et al., 2025]. Under standard caching, the reuse decision is made based on a single prediction, because its cache-induced error directly enters the denoising update. Under CFG caching, existing methods often extend the same idea to the conditional and unconditional branches, using branch-wise changes as reuse signals. However, branch-local reuse criteria do not explicitly account for how the two branch errors interact after CFG composition.

As illustrated in Fig. 1, under CFG the denoising update is determined by the guided composition of the conditional and unconditional predictions. Consequently, the cache-induced perturbation relevant to sampling is the error after CFG composition rather than the error of either branch in isolation. We refer to this discrepancy as the *branch–guided mismatch*: branch-error magnitudes alone do not fully characterize the guided error that drives the denoising update. Because the guided error depends on both the magnitudes and relative directions of the two branch errors under the guidance scale, their interaction affects the resulting perturbation. As shown in Fig. 2(a), the guided-error curve can differ substantially from the two branch-wise error curves, especially in the early stage, showing that branch-wise error magnitudes need not directly track the guided-error magnitude. We formally characterize this relationship through the CFG error geometry in Section 3.2.

Accounting for the *branch–guided mismatch* alone is still insufficient, because the downstream effect of a local perturbation also depends on when it is introduced. Prior work has examined downstream cache effects through outcome-aware scheduling and error rectification [Gao et al., 2025, Peng et al., 2026]. Here, we study this effect together with CFG branch-error composition. We demonstrate it through a single-step perturbation study in Fig. 2(b): cache reuse is applied only at a specific timestep, while all other steps use full computation. We observe that similar local guided errors can lead to substantially different final deviations depending on the perturbed timestep. We refer to this discrepancy as the *local–final mismatch*. This suggests that cache control should account not only for

the local guided error, but also for its downstream impact on the final sample. We further quantify this effect in Section 3.3.

Motivated by these observations, we propose **RA-CFGCache**, a **R**isk-**A**ligned **Caching** framework for **CFG**. Based on the above analysis, cache reuse under CFG should be governed by a risk estimate aligned with the cache-induced perturbation on the guided prediction, while also accounting for how this perturbation propagates to the final sample. *CFG-aware Guided-Risk Composition* combines existing branch-wise proxies using the CFG composition structure and calibrated cross-branch alignment to construct a local guided-risk estimate, thereby addressing the *branch–guided mismatch*. *Propagation-Aware Rescaling* then weights this risk estimate using a lightweight timestep-dependent propagation prior to account for the downstream impact of local perturbations, thereby addressing the *local–final mismatch*. The resulting risk estimate is used by a lightweight online threshold controller to govern joint refresh and reuse of the two branches, while keeping the sampling schedule and guidance rule unchanged.

Our contributions are summarized as follows:

- **Characterizing two key misalignments under CFG caching.** We show that cache control under CFG is affected by two distinct sources of misalignment: the *branch–guided mismatch*, where branch-wise errors do not fully characterize the guided error that drives the denoising update, and the *local–final mismatch*, where the downstream impact of a local guided error varies across timesteps.
- **Introducing a risk-aligned caching framework for CFG.** We propose **RA-CFGCache**, which addresses these two misalignments through two coupled designs: *CFG-aware guided-risk composition*, which combines branch-wise proxies using the CFG composition structure and calibrated cross-branch alignment to construct a local guided-risk estimate, and *propagation-aware rescaling*, which weights this risk estimate using a timestep-dependent propagation prior to account for the downstream impact of local guided errors.
- **Demonstrating improved efficiency–fidelity trade-offs under CFG.** Experiments and ablations on diffusion transformer pipelines show that **RA-CFGCache** improves the efficiency–fidelity trade-off over the evaluated training-free caching baselines under CFG. Risk diagnostics and controlled comparisons further support the proposed analysis and control strategy.

2 Related Work

Training-free diffusion caching. Training-free diffusion caching accelerates diffusion inference by reusing intermediate features, residuals, or model outputs across timesteps without retraining. Existing methods broadly follow three overlapping directions. The first follows *pre-defined reuse*, where computation is reused according to fixed layer, block, or timestep schedules, as exemplified by DeepCache Ma et al. [2023], FasterDiffusion Li et al. [2024], PAB Zhao et al. [2025], FORA Selvaraju et al. [2024], and Δ-DiT Chen et al. [2024]. The second follows *adaptive reuse*, where control signals based on temporal change, residual magnitude, token importance, sensitivity, or estimated output variation determine when or where cached computation can be reused, as in TeaCache Liu et al. [2025a], MagCache Ma et al. [2025], ToCa Zou et al. [2025], DiCache Bu et al. [2025], and SenCache Haghighi and Alahi [2026]. A third direction shifts *from reuse to prediction*, estimating future features or outputs through extrapolation, interpolation, or empirical correction, as in TaylorSeer Liu et al. [2025b], HiCache Feng et al. [2026], and FoCa Zheng et al. [2025]. Recent methods also consider downstream cache effects: LeMiCa Gao et al. [2025] uses final-output impact for global error-aware scheduling, while ERTACache Peng et al. [2026] analyzes cache-error propagation and introduces corresponding correction mechanisms. In contrast, **RA-CFGCache** studies downstream cache effects together with CFG-specific branch-error composition, combining cross-branch error interaction and timestep-dependent propagation weighting in a unified criterion.

CFG-aware acceleration and guidance optimization. Classifier-free guidance (CFG) Ho and Salimans [2022] improves conditional generation quality, but explicit two-branch CFG requires both conditional and unconditional evaluations. Existing approaches exploit *cross-branch redundancy*, modify the *guidance process*, or jointly optimize *guidance and caching*. FasterCache Lv et al. [2025] exploits redundancy between conditional and unconditional features and introduces CFG-Cache for branch reuse. Guidance optimization methods include per-timestep guidance scheduling

such as CFG Schedulers Wang et al. [2024], interval-based strategies such as Apply Guidance in Interval Kynkäänniemi et al. [2024], and instance-aware magnitude control such as MAMBO-G Zhu et al. [2026]. OUSAC Sun et al. [2025] further jointly optimizes CFG timesteps, guidance scales, and caching. In contrast, **RA-CFGCache** keeps the sampling schedule and guidance rule unchanged and controls joint branch refresh and reuse through a guided-risk estimate that accounts for both cross-branch error interaction and downstream propagation.

3 Method

3.1 Preliminaries

We consider conditional diffusion / flow-based generative models implemented with diffusion transformers (DiTs), where inference proceeds through an iterative denoising process. At each timestep t, the model predicts from the current latent x_t and the conditioning input, and the sampler updates the latent state using this prediction. We denote the prediction function by $f_\theta(\cdot)$, where θ denotes the fixed model parameters.

Training-free caching accelerates inference by reusing cached computation from a previous timestep instead of recomputing it at the current one. Although different methods may cache predictions, residuals, or intermediate features, their effect on the sampler can be abstracted as replacing a fresh prediction p_t with the actually used prediction $\tilde{p}_t$. Both are defined at the same current latent x_t: p_t is obtained by full computation, whereas $\tilde{p}_t$ is obtained with cache reuse. The resulting local cache-induced error is

$$\delta p_t = \tilde{p}_t - p_t. \tag{1}$$

Here, p_t serves as an analytical reference and need not be evaluated by the online controller.

Under classifier-free guidance (CFG) [Ho and Salimans, 2022], the model evaluates both unconditional and conditional branches:

$$u_t = f_\theta(x_t, t, \emptyset), \qquad c_t = f_\theta(x_t, t, y), \tag{2}$$

where u_t and c_t denote the unconditional and conditional predictions, respectively. The prediction entering the sampler is

$$p_t^{(s)} = u_t + s(c_t - u_t), \tag{3}$$

where s is the guidance scale. This is equivalent to the original CFG formulation of Ho and Salimans [2022] up to a reparameterization of the guidance weight. Therefore, under CFG, the denoising update is governed by the guided prediction $p_t^{(s)}$ rather than by either branch alone. Although reuse is applied at the branch level, its local effect on sampling is determined by the induced error on the guided prediction.

3.2 Branch–Guided Mismatch

Under CFG, cache reuse affects the denoising update through the guided prediction rather than through either branch alone. We formalize this by explicitly characterizing the guided error induced by branch-level reuse.

Let u_t and c_t denote the fresh unconditional and conditional branch predictions defined in Eq. (2). Under cache reuse, the actual branch predictions used in guidance composition are

$$\tilde{u}_t = u_t + \delta u_t, \qquad \tilde{c}_t = c_t + \delta c_t, \tag{4}$$

where $\delta u_t, \delta c_t$ are the corresponding cache-induced branch errors. The resulting guided prediction under reuse is

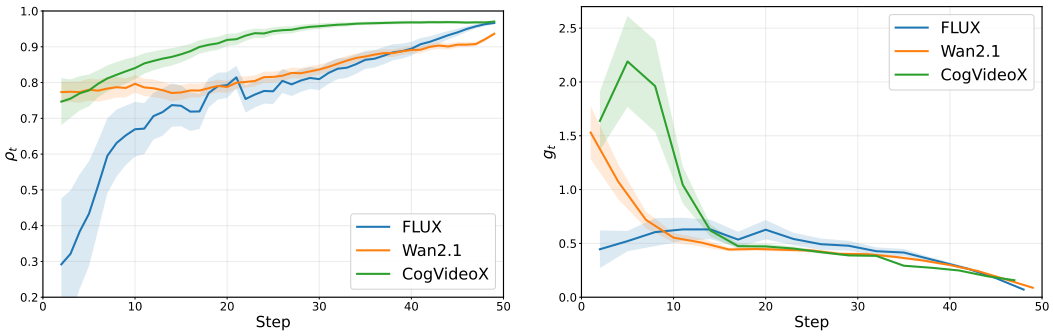
$$\tilde{p}_t^{(s)} = \tilde{u}_t + s(\tilde{c}_t - \tilde{u}_t), \tag{5}$$

so the cache-induced guided error becomes

$$\delta p_t = \tilde{p}_t^{(s)} - p_t^{(s)} = (1-s)\delta u_t + s\delta c_t. \tag{6}$$

Let $\rho_t = \cos(\delta u_t, \delta c_t)$ denote the cosine similarity between the two branch errors. When either error is zero, we set $\rho_t = 0$ since the cross-term vanishes. Then

$$\|\delta p_t\|_2^2 = (1-s)^2\|\delta u_t\|_2^2 + s^2\|\delta c_t\|_2^2 + 2s(1-s)\|\delta u_t\|_2\|\delta c_t\|_2\rho_t. \tag{7}$$



(a) **Branch-error alignment** (b) **Propagation gain**

Figure 3: **Two sources of mismatch under CFG caching.** (a) The alignment $\rho_t = \cos(\delta u_t, \delta c_t)$ varies across timesteps, inducing timestep-dependent cancellation or amplification in Eq. (7). (b) The propagation gain g_t varies across timesteps, showing that similar local guided errors can induce different final deviations. Solid curves show the mean over 100 prompts, and shaded bands indicate $\pm$ one standard deviation across prompts.

Equation (7) shows that guided error depends not only on branch-error magnitudes, but also on their alignment through the cross-term. Under the practical CFG regime $s > 1$, the coefficient $2s(1-s)$ is negative, so positive alignment induces partial cancellation, whereas weak or negative alignment suppresses this effect and can even lead to amplification. As shown in Fig. 3a, this alignment is timestep-dependent: the cancellation effect is weaker in earlier steps and stronger in later ones. We provide a more detailed analysis of this CFG error geometry and the origin of the alignment statistic in Appendix C.1. Consequently, branch-wise error no longer faithfully reflects the true local perturbation under CFG. This constitutes the *branch–guided mismatch*.

3.3 Local–Final Mismatch

Even guided error is only a local quantity at the current timestep. For cache control, what ultimately matters is its effect on the final generated sample after subsequent denoising steps.

To isolate this effect, we consider a single-step perturbation setting. The perturbed and reference trajectories are identical before timestep t; cache reuse is applied only at timestep t, while all other steps use full computation under the same initial noise and conditioning. The local guided error is defined as

$$e_t^{\text{local}} = \|\delta p_t\|_2. \tag{8}$$

The resulting final deviation is measured by

$$e_t^{\text{final}} = \|x_0^{(t)} - x_0^{\text{full}}\|_2, \tag{9}$$

where $x_0^{(t)}$ and x_0^{full} denote the final latent states of the perturbed and full-computation trajectories, respectively. For $e_t^{\text{local}} > 0$, we then define the propagation gain as

$$g_t = \frac{e_t^{\text{final}}}{e_t^{\text{local}}}. \tag{10}$$

We provide additional details of propagation analysis in Appendix C.2.

Figure 3(b) shows that the propagation gain varies systematically across timesteps: similar local guided errors can induce different final deviations depending on when they occur. We refer to this as the *local–final mismatch*, motivating a timestep-dependent propagation prior for cache control.

3.4 RA-CFGCache

RA-CFGCache realizes cache control under CFG as *guided-risk control*, as illustrated in Fig. 4. It contains two components: *CFG-aware guided-risk composition*, which constructs a CFG-informed guided-risk estimate from branch-wise proxies, and *propagation-aware rescaling*, which weights this risk estimate using a calibrated timestep-dependent propagation prior.

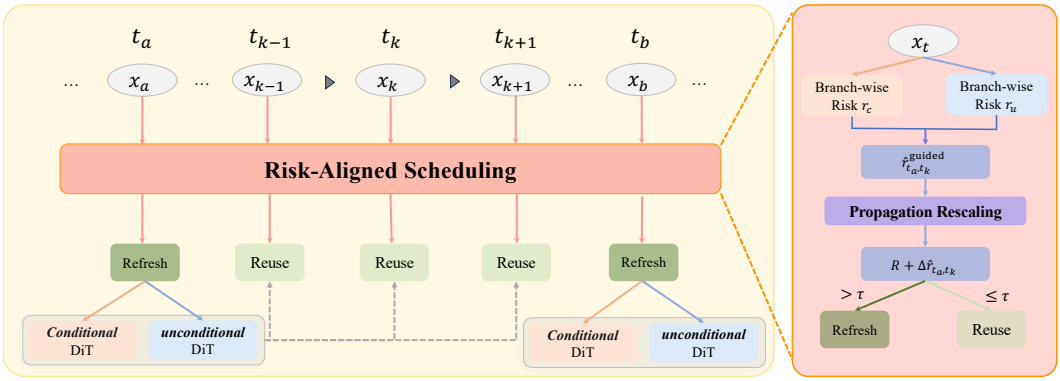


Figure 4: **Overview of RA-CFGCache.** Under classifier-free guidance (CFG), cache control should be aligned with the guided prediction rather than either branch alone. **RA-CFGCache** first combines branch-wise proxy estimates into a guided-risk estimate, then rescales it using a timestep-dependent propagation prior, and finally uses the resulting risk estimate for online refresh decisions.

CFG-aware Guided-Risk Composition. Since true branch errors are unavailable online, we start from nonnegative branch-wise proxy estimates provided by a base caching method, such as TeaCache [Liu et al., 2025a], MagCache [Ma et al., 2025], or DiCache [Bu et al., 2025]. Let $\hat{r}_u(t_a, t_b)$ and $\hat{r}_c(t_a, t_b)$ denote the proxy estimates for the current candidate reuse from anchor timestep t_a to current timestep t_b; accumulation is handled separately by the controller below. The exact proxy definitions used in our experiments are provided in Appendix B.2. Motivated by the exact CFG error decomposition in Eq. (7), we combine them using the same quadratic form:

$$\left(\hat{r}^{\text{guided}}_{t_a,t_b}\right)^2 = (1-s)^2\hat{r}_u^2(t_a,t_b) + s^2\hat{r}_c^2(t_a,t_b) + 2s(1-s)\,\bar{\rho}_{t_a,t_b}\,\hat{r}_u(t_a,t_b)\hat{r}_c(t_a,t_b), \tag{11}$$

and take the nonnegative square root to obtain $\hat{r}^{\text{guided}}_{t_a,t_b}$. Here, $\bar{\rho}_{t_a,t_b}$ is an offline estimate of cross-branch alignment. On a held-out calibration split, we compute the cosine similarity between conditional and unconditional temporal differences of the full-computation residual outputs for each pair (t_a, t_b) and average it across samples. We refer to $\hat{r}^{\text{guided}}_{t_a,t_b}$ as a proxy-based guided-risk estimate used for cache control, rather than an exact estimate of the true guided-error magnitude.

Propagation-aware Rescaling. To account for non-uniform error propagation, we further rescale the guided-risk estimate using a timestep-dependent propagation prior. The single-step perturbation study reveals systematic variation across timesteps, which we summarize with a lightweight power-law surrogate:

$$\hat{g}_k = \lambda\left(\frac{T-k}{T}\right)^{\alpha} + \beta, \tag{12}$$

where k is the increasing sampler-step index, T is the total number of steps, and λ, α, β are fitted from the calibration measurements. The power-law form is empirical; fitting details are provided in Appendix D.6. Here, a and b denote the sampler-step indices corresponding to timesteps t_a and t_b. The propagation-aware risk contribution is

$$\Delta\hat{r}_{t_a,t_b} = \max\left(0, \hat{g}_b\,\hat{r}^{\text{guided}}_{t_a,t_b}\right). \tag{13}$$

This quantity incorporates both CFG error composition and the timestep-dependent downstream effect of local perturbations.

Online Control Policy. We maintain an accumulated-risk score R within each reuse interval. At the beginning of an interval, the anchor step is set to t_a and R is initialized to zero. At the current step t_b, the controller computes $\Delta\hat{r}_{t_a,t_b}$ in Eq. (13). A refresh is triggered when

$$R + \Delta\hat{r}_{t_a,t_b} > \tau, \tag{14}$$

where τ controls the reuse–refresh trade-off. Otherwise, reuse continues and

$$R \leftarrow R + \Delta\hat{r}_{t_a,t_b}. \tag{15}$$

After a refresh, the anchor is reset to the current step and R is reinitialized to zero. The accumulated-risk score is an interval-level control quantity rather than an exact estimate or formal bound on the final output deviation.

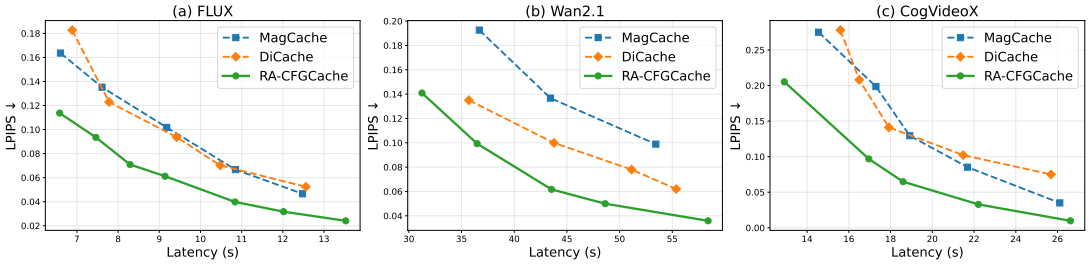


Figure 5: **Trade-off curves. RA-CFGCache** achieves lower LPIPS at comparable latency across models.

Table 1: **Main results on text-to-image and text-to-video generation.** Comparison of **RA-CFGCache** with recent diffusion acceleration baselines.

Model	Method	LPIPS ↓	SSIM ↑	PSNR ↑	Speedup ↑	Latency (s) ↓
FLUX.1-dev	Vanilla ($T = 50$)	0.0000	1.0000	∞	1.00×	21.02
	Vanilla ($T = 25$)	0.4556	0.6207	13.09	1.96×	10.71
	TeaCache (τ=0.4)	0.4471	0.6174	13.80	2.53×	8.31
	TaylorSeer ($\mathcal{N}$=3,$\mathcal{O}$=1)	0.4379	0.6398	13.75	2.37×	8.86
	DiCache ($m = 2, \tau$=0.4)	0.1431	0.8229	22.13	2.94×	7.15
	MagCache (τ=0.24)	0.1636	0.8177	21.46	3.18×	6.60
	HiCache ($\mathcal{N}$=3,$\mathcal{O}$=1)	0.3937	0.6646	14.72	2.35×	8.94
	FasterCache ($I_\Delta = 8, I_{\text{attn}} = 8$)	0.2158	0.7593	20.88	2.09×	10.06
	RA-CFGCache-Slow (τ=0.18)	**0.1137**	**0.8680**	**24.02**	3.19×	6.58
	RA-CFGCache-Fast (τ=0.3)	0.1484	0.8340	22.75	**3.95**×	**5.32**
Wan2.1-T2V-1.3B	Vanilla ($T = 50$)	0.0000	1.0000	∞	1.00×	93.32
	Vanilla ($T = 25$)	0.4371	0.5658	15.87	1.93×	48.41
	TeaCache (τ=0.15)	0.1661	0.7955	23.08	2.44×	38.24
	TaylorSeer ($\mathcal{N}$=3,$\mathcal{O}$=1)	0.3632	0.6187	16.96	2.13×	43.79
	DiCache ($m = 2, \tau$=0.4)	0.1350	0.8361	24.90	2.61×	35.70
	MagCache (τ=0.1)	0.1926	0.7696	22.16	2.54×	36.69
	HiCache ($\mathcal{N}$=3,$\mathcal{O}$=1)	0.2916	0.6711	18.79	2.14×	43.49
	FasterCache ($I_\Delta = 8, I_{\text{attn}} = 8$)	0.3483	0.6532	20.37	2.41×	38.71
	RA-CFGCache-Slow (τ=0.4)	**0.0993**	**0.8778**	**28.03**	2.56×	36.48
	RA-CFGCache-Fast (τ=0.6)	0.1409	0.8386	26.12	**2.99**×	**31.25**
CogVideoX-2B	Vanilla ($T = 50$)	0.0000	1.0000	∞	1.00×	40.48
	Vanilla ($T = 25$)	0.4751	0.5445	13.61	1.86×	21.79
	TeaCache (τ=0.2)	0.2040	0.7538	21.48	2.42×	16.72
	DiCache ($m = 2, \tau$=0.4)	0.2080	0.7710	22.35	2.44×	16.57
	MagCache (τ=0.08)	0.2386	0.7230	20.33	2.34×	17.30
	FasterCache ($I_\Delta = 6, I_{\text{attn}} = 6$)	0.1157	0.8573	25.48	2.03×	19.94
	RA-CFGCache-Slow (τ=0.6)	**0.0968**	**0.8746**	**26.93**	2.39×	16.96
	RA-CFGCache-Fast (τ=1.4)	0.2052	0.7614	21.89	**3.13**×	**12.91**

Reuse Action. We adopt joint reuse of both CFG branches as the default action, allowing the controller to account for their combined proxy risk. When branch errors are positively aligned and $s > 1$, their contributions can partially cancel under CFG composition. Joint reuse also skips recomputation of the cached components in both branches. Under parallel branch execution, one-sided reuse may provide limited latency savings because the guided prediction must still wait for the recomputed branch. Detailed comparisons are provided in Appendix C.3 and Appendix D.7.

4 Experiments

4.1 Experimental Setup

Models and baselines. We evaluate our method on diffusion transformer pipelines under classifier-free guidance, including FLUX.1-dev [Labs, 2024] for text-to-image generation, Wan2.1-T2V-1.3B [Wan et al., 2025] and CogVideoX-2B [Yang et al., 2025] for text-to-video generation. We



Figure 6: **Qualitative comparison on FLUX.1-dev.** We show three representative text-to-image examples under the same prompts and random seeds. Compared with TeaCache, MagCache, and DiCache, **RA-CFGCache** better preserves local textures, object structures, and semantic details under comparable acceleration settings.

Table 2: **Contribution of each component.** Each component improves the trade-off, and combining both gives the best result.

CFG-aware Composition	Propagation Rescaling	Speedup↑	LPIPS↓	SSIM↑	PSNR↑
✗	✗	3.42×	0.1732	0.8076	21.12
✓	✗	3.41×	0.1391	0.8462	22.82
✗	✓	3.41×	0.1423	0.8354	22.47
✓	✓	**3.43×**	**0.1262**	**0.8565**	**23.45**

Table 3: **Effect of the reused quantity.** Joint branch reuse gives the best overall result under a matched refresh budget.

Reused quantity	Speedup↑	LPIPS↓	SSIM↑	PSNR↑
Always refresh	1.00×	0.0000	1.0000	∞
Reuse unconditional only	1.55×	0.2943	0.7281	19.80
Reuse conditional only	1.47×	0.2317	0.7630	18.62
Reuse Δ only	1.55×	0.1939	0.7940	19.91
Reuse both branches	**3.38×**	**0.1729**	**0.8112**	**21.17**

compare against representative training-free acceleration baselines, including TeaCache [Liu et al., 2025a], TaylorSeer [Liu et al., 2025b], DiCache [Bu et al., 2025], MagCache [Ma et al., 2025], HiCache [Feng et al., 2026], and the CFG-related FasterCache [Lv et al., 2025]. All methods use the same models, prompts, denoising schedules, and hardware unless otherwise specified.

Main evaluation settings. Unless otherwise specified, all main experiments use 50 denoising steps. For FLUX.1-dev, we generate 1024×1024 images with true CFG scale and guidance embedding both set to 3.5. For Wan2.1-T2V-1.3B, we generate 832×480 videos with 81 frames, 16 FPS and guidance scale 5.0. For CogVideoX-2B, we generate 720×480 videos with 49 frames, 8 FPS and guidance scale 6.0.

Data and metrics. For text-to-image evaluation, we use 200 DrawBench prompts [Saharia et al., 2022]. For text-to-video evaluation, we use 100 prompts sampled from the `all_dimension.txt` prompt set of VBench [Huang et al., 2023]. Unless otherwise specified, all accelerated outputs are compared against the corresponding vanilla outputs under the same prompt and random seed. We report end-to-end latency and speedup as efficiency metrics, and LPIPS [Zhang et al., 2018], SSIM [Wang et al., 2004], and PSNR against vanilla outputs as fidelity metrics.

Implementation details. All experiments are implemented in PyTorch on NVIDIA H200 GPUs, primarily using bfloat16 with selected operations in float32. Each sample uses one GPU, with the two CFG branches evaluated sequentially in the main experiments. **RA-CFGCache** uses the threshold-based scheduler in Section 3.4 with joint branch reuse by default. Unless otherwise specified, it uses a MagCache-style proxy for FLUX.1-dev and a TeaCache-style proxy for Wan2.1-T2V-1.3B and CogVideoX-2B. We report **RA-CFGCache-Fast** and **RA-CFGCache-Slow** using different

Table 4: **Robustness across CFG scales. RA-CFGCache** maintains a strong efficiency–fidelity trade-off across different guidance scales.

CFG	Method	Speedup↑	LPIPS↓	SSIM↑	PSNR↑
1.5	DiCache	3.11×	0.2042	0.8114	22.55
	MagCache	3.18×	0.1682	0.8268	22.42
	FasterCache	2.09×	0.2002	0.7814	21.58
	Ours	**3.38×**	**0.1430**	**0.8553**	**24.08**
3.5	DiCache	2.94×	0.1431	0.8229	22.13
	MagCache	3.18×	0.1636	0.8177	21.46
	FasterCache	2.09×	0.2158	0.7593	20.88
	Ours	**3.43×**	**0.1262**	**0.8565**	**23.45**
5.5	DiCache	2.91×	0.1378	0.8460	24.08
	MagCache	3.18×	0.1558	0.8423	23.13
	FasterCache	2.01×	0.2710	0.6969	19.11
	Ours	**3.35×**	**0.1153**	**0.8779**	**25.30**
7.5	DiCache	2.88×	0.1163	0.8692	25.65
	MagCache	3.18×	0.1295	0.8673	24.81
	FasterCache	2.00×	0.2784	0.6695	19.34
	Ours	**3.37×**	**0.1031**	**0.8867**	**26.41**

Table 5: **Validation of Guided-Risk Composition.** The full formulation improves guided-error alignment and high-risk reuse detection under an identical refresh budget.

Variant	Pearson↑	AUPRC↑	P95↓	P99↓
No-cross	0.6082	0.5671	0.3247	0.5536
Full Eq. 11	0.7575	0.6799	0.2809	0.4635
Proxy mag. + true align.	0.8325	0.7601	0.2662	0.4145
True mag. + est. align.	0.9358	0.9178	0.1976	0.2840
Oracle geometry	1.0000	1.0000	0.1744	0.1874

Table 6: **Validation of Propagation-Aware Rescaling.** Propagation weighting improves configuration-level prediction of mean final LPIPS under repeated reuse.

Predictor	Spearman↑	LOO R^2 ↑	LOO MAE↓
Unweighted	0.6571	−0.1773	0.0248
Prop.-weighted	**0.9429**	**0.5274**	**0.0157**

risk thresholds τ. For each model, offline calibration estimates the pairwise alignment statistic $\bar{\rho}_{t_a,t_b}$ and timestep-dependent rescaling coefficients, which are fixed during evaluation. Reported latency includes online cache-control overhead but excludes offline calibration. Additional details are provided in Appendix B.

4.2 Main Results

Quantitative Comparison. Table 1 reports the main quantitative results under CFG. Across the evaluated text-to-image and text-to-video pipelines, **RA-CFGCache** achieves favorable efficiency–fidelity trade-offs relative to the evaluated baselines. At comparable latency, it improves fidelity over strong caching baselines, while its fast variants provide additional acceleration with a corresponding fidelity trade-off.

Trade-off curves. Figure 5 compares latency–LPIPS trade-offs across FLUX.1-dev, Wan2.1-T2V-1.3B, and CogVideoX-2B. Across the evaluated operating range, **RA-CFGCache** achieves lower LPIPS at comparable latency than MagCache and DiCache. The improvement across multiple operating points shows that the gain is not limited to a single tuned threshold.

Qualitative comparison. Figure 6 compares **RA-CFGCache** with TeaCache, MagCache, and DiCache on FLUX.1-dev at comparable speedups. The baselines introduce visible scene changes or local structural artifacts in the highlighted regions, while **RA-CFGCache** remains closer to the original samples and better preserves textures, structures, and semantic details. Additional qualitative results on Wan2.1 and CogVideoX are provided in Appendix F.

4.3 Ablation Studies

Unless otherwise specified, all ablations are conducted on FLUX.1-dev under the main experimental setting. We examine the contributions of the proposed components and reuse policy.

Contribution of each component. We ablate the two components of **RA-CFGCache**: CFG-aware guided-risk composition and propagation-aware rescaling. As shown in Table 2, each component improves fidelity at similar latency, and their combination performs best. These results support their complementary benefits in the evaluated setting.

Effect of the reused quantity. We compare different reuse targets under the same refresh count. As shown in Table 3, joint branch reuse achieves the best fidelity among the evaluated actions and larger latency savings by skipping recomputation in both branches. A separate parallel-execution study in Appendix D.7 compares independent branch-wise and joint controllers at matched branch-evaluation counts, showing lower latency for joint control. Appendix C.3 further analyzes reuse-object selection through CFG error geometry and computational savings.

Table 7: **Robustness of calibration on $\bar{\rho}_{t_a,t_b}$.** Downstream fidelity and speedup remain relatively stable across the tested calibration-set sizes and prompt subsets.

Strategy	#	LPIPS↓	SSIM↑	PSNR↑	Speedup↑
Random	1	0.1345	0.8472	23.01	3.47×
Random	10	0.1301	0.8507	23.28	3.46×
Outlier	10	0.1326	0.8495	23.19	3.45×
Random	50	0.1308	0.8500	23.26	3.46×
Random	100	**0.1297**	**0.8513**	**23.29**	3.46×

Table 8: **Robustness of calibration on g_t.** Downstream fidelity and speedup remain relatively stable across the tested calibration-set sizes and prompt subsets.

Strategy	#	LPIPS↓	SSIM↑	PSNR↑	Speedup↑
Random	1	0.1426	0.8380	22.94	3.46×
Random	10	0.1329	0.8497	23.21	3.46×
Outlier	10	0.1448	0.8379	22.80	3.46×
Random	50	0.1297	0.8513	23.29	3.46×
Random	100	**0.1262**	**0.8579**	**23.80**	3.46×

Effect of CFG scale. Table 4 shows that **RA-CFGCache** achieves a strong efficiency–fidelity trade-off across different guidance scales. Across all tested CFG scales, it remains among the best methods in trade-off, while maintaining stable speedup and strong fidelity relative to competing baselines. This indicates that risk-aligned cache control remains effective across the evaluated guidance strengths.

Validation of risk composition and propagation rescaling. Table 5 evaluates guided-risk composition on 37,400 reuse events from 50 held-out FLUX.1-dev prompts disjoint from offline calibration. Pearson correlation measures agreement with true guided-error magnitude, while AUPRC measures detection of the top 20% highest-error events. Under the same 30% refresh budget, P95 and P99 report true-error percentiles among accepted reuse events. The full composition improves correlation and high-risk discrimination while reducing these tail errors. Oracle-assisted variants substitute true branch-error magnitudes and/or alignment for diagnosis only and are unavailable online.

Table 6 evaluates propagation weighting on six configurations: three thresholds $\tau \in \{0.12, 0.18, 0.30\}$, each with and without rescaling, using 50 held-out prompts per configuration. For each trajectory, we sum the unweighted guided-risk contributions or their propagation-weighted counterparts over all accepted reuse steps, including across refresh intervals. We then average each score and final LPIPS over prompts. Spearman correlation is computed across the six configuration means. For R^2 and MAE, we use leave-one-configuration-out affine regression: each configuration's mean LPIPS is predicted from its mean risk using a model fitted on the other five configurations. Propagation weighting increases Spearman correlation from 0.6571 to 0.9429 and improves held-out R^2 and MAE, supporting its predictive value at the configuration level.

Validation of alignment and propagation calibration. We further evaluate the robustness of the calibrated quantities by varying only the calibration prompt subset while keeping the evaluation protocol and controller unchanged. As shown in Tables 7 and 8, downstream fidelity and speedup remain relatively stable across the tested random and outlier calibration subsets and calibration-set sizes. These results indicate robustness to the tested calibration-set variations, rather than invariance of the calibrated alignment or propagation curves themselves.

5 Conclusion

We revisit training-free diffusion caching under CFG as guided-error control. **RA-CFGCache** aligns reuse decisions with the guided prediction error rather than branch-local criteria, and further rescales this risk by its timestep-dependent propagation to the final sample. By combining CFG-aware guided-risk composition, propagation-aware rescaling, and threshold-based scheduling, **RA-CFGCache** consistently improves the efficiency–fidelity trade-off over the evaluated training-free caching baselines on text-to-image and text-to-video diffusion transformers.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62525103, 62271281 and 624B2082) and the Beijing Natural Science Foundation (Grant No. L247026). The authors declare no competing financial interests.

References

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. URL https://arxiv.org/abs/1907.05600.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023. URL https://arxiv.org/abs/2210.03142.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023. URL https://arxiv.org/abs/2311.17042.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

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, 2022. URL https://arxiv.org/abs/2206.00927.

Evelyn Zhang, Bang Xiao, Jiayi Tang, Qianli Ma, Chang Zou, Xuefei Ning, Xuming Hu, and Linfeng Zhang. Token pruning for caching better: 9 times acceleration on stable diffusion for free, 2024. URL https://arxiv.org/abs/2501.00375.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free, 2023. URL https://arxiv.org/abs/2312.00858.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7353–7363, June 2025a.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. Magcache: Fast video generation with magnitude-aware cache. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KZn7TDOL4J.

Huanlin Gao, Ping Chen, Fuyuan Shi, Chao Tan, Zhaoxiang Liu, Fang Zhao, Kai Wang, and Shiguo Lian. Lemica: Lexicographic minimax path caching for efficient diffusion-based video generation. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2511.00090.

Xurui Peng, Chenqian Yan, Hong Liu, Rui Ma, Fangmin Chen, Xing Wang, Zhihua Wu, Songwei Liu, and Mingbao Lin. Ertacache: Error rectification and timesteps adjustment for efficient diffusion. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=InvyBiYcK5.

Senmao Li, Taihang Hu, Joost van de Weijer, Fahad Shahbaz Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of the encoder for diffusion model inference, 2024. URL https://arxiv.org/abs/2312.09608.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast, 2025. URL https://arxiv.org/abs/2408.12588.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration, 2024. URL https://arxiv.org/abs/2407.01425.

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. δ-dit: A training-free acceleration method tailored for diffusion transformers, 2024. URL https://arxiv.org/abs/2406.01125.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching, 2025. URL https://arxiv.org/abs/2410.05317.

Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. Dicache: Let diffusion model determine its own cache, 2025. URL https://arxiv.org/abs/2508.17356.

Yasaman Haghighi and Alexandre Alahi. Sencache: Accelerating diffusion model inference via sensitivity-aware caching, 2026. URL https://arxiv.org/abs/2602.24208.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15853–15863, October 2025b.

Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, and Linfeng Zhang. Hicache: A plug-in scaled-hermite upgrade for taylor-style cache-then-forecast diffusion acceleration, 2026. URL https://arxiv.org/abs/2508.16984.

Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers, 2025. URL https://arxiv.org/abs/2508.16211.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality, 2025. URL https://arxiv.org/abs/2410.19355.

Xi Wang, Nicolas Dufour, Nefeli Andreou, Marie-Paule Cani, Victoria Fernandez Abrevaya, David Picard, and Vicky Kalogeiton. Analysis of classifier-free guidance weight schedulers, 2024. URL https://arxiv.org/abs/2404.13040.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models, 2024. URL https://arxiv.org/abs/2404.07724.

Shangwen Zhu, Qianyu Peng, Zhilei Shu, Yuting Hu, Zhantao Yang, Han Zhang, Zhao Pu, Andy Zheng, Xinyu Cui, Jian Zhao, Ruili Feng, and Fan Cheng. Mambo-g: Magnitude-aware mitigation for boosted guidance, 2026. URL https://arxiv.org/abs/2508.03442.

Ruitong Sun, Tianze Yang, Wei Niu, and Jin Sun. Ousac: Optimized guidance scheduling with adaptive caching for dit acceleration, 2025. URL https://arxiv.org/abs/2512.14096.

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025. URL https://arxiv.org/abs/2408.06072.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models, 2023. URL https://arxiv.org/abs/2311.17982.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Appendix

A Limitations

While **RA-CFGCache** shows consistent gains under the evaluated classifier-free guidance settings, the current formulation still has several methodological limitations.

First, the propagation gain is an empirical timestep-dependent surrogate motivated by first-order propagation analysis, rather than a complete model of how local guided error affects the final output. It is estimated from isolated single-step perturbations and therefore does not explicitly model higher-order interactions among errors introduced by consecutive reuse decisions.

Second, although the cumulative scheduler goes beyond purely local thresholding, it remains an online approximation to the underlying sequential control problem and does not explicitly optimize long-horizon interactions among reuse decisions or provide a formal optimality guarantee.

Third, the alignment and propagation statistics rely on offline calibration for a given inference configuration, and their transfer across substantially different models, samplers, timestep schedules, or guidance settings may require validation or recalibration. Improving such transferability while retaining lightweight online control is an important direction for future work.

B Implementation Details

B.1 Details of the Compared Methods

TeaCache [Liu et al., 2025a] is a training-free, architecture-agnostic caching method that exploits the correlation between timestep embedding changes and model output differences across adjacent steps. It activates caching through an accumulated error-based thresholding strategy. We adapt its official implementation to our evaluation repository and use model-specific representative thresholds as reported in Table 1: $\tau = 0.4$ for FLUX.1-dev, $\tau = 0.15$ for Wan2.1-T2V-1.3B, and $\tau = 0.2$ for CogVideoX-2B.

TaylorSeer [Liu et al., 2025b] is a training-free acceleration method based on the "cache-then-forecast" paradigm. Instead of directly reusing cached features, it predicts future features using Taylor-series-based extrapolation. We adapt the released implementation and use the reported configuration with activation/cache interval $N = 3$ and Taylor expansion order $O = 1$.

DiCache [Bu et al., 2025] is a training-free caching method that reuses diffusion outputs through a lightweight temporal reuse policy. We adapt its official implementation and use probe depth $m = 2$ and threshold $\tau = 0.4$ across the reported models.

MagCache [Ma et al., 2025] controls acceleration through a maximum skip length and an accumulated error threshold. We adapt its official implementation, use the default fast-mode maximum skip length $K = 4$, and use model-specific thresholds as reported in Table 1: $\tau = 0.24$ for FLUX.1-dev, $\tau = 0.1$ for Wan2.1-T2V-1.3B, and $\tau = 0.08$ for CogVideoX-2B. The magnitude-ratio lists are obtained from the official repository or the corresponding official calibration procedure in `diffusers`.

HiCache [Feng et al., 2026] is a higher-order caching method that improves temporal reuse using a scaled Hermite-style update. We adapt the released implementation and use cache interval $\mathcal{N} = 3$ and update order $\mathcal{O} = 1$.

FasterCache [Lv et al., 2025] is a training-free CFG acceleration method that exploits redundancy between the conditional and unconditional branches and reuses self-attention computations. We use the released implementation for CogVideoX and backend-specific adapters for FLUX and Wan, without changing the model, sampling steps, resolution, or guidance. For FLUX and Wan, we set $I_{\Delta} = 8$ and $I_{\mathrm{attn}} = 8$, use an extrapolation coefficient of 0.3 and $t_0 = 0.5$, and disable frequency gains. For CogVideoX, we retain the released frequency compensation, set $I_{\Delta} = I_{\mathrm{attn}} = 6$, and keep the extrapolation coefficient at 0.3. Configurations are selected using calibration prompts only and frozen for evaluation; the FLUX configuration is transferred to Wan without further tuning.

We additionally include a reduced-step vanilla baseline, **Vanilla** $(T = 25)$, which uses the same model and sampling configuration as the full vanilla baseline except for halving the denoising steps from $T = 50$ to $T = 25$. For all baselines, we use the same model checkpoints, prompts, random seeds, and hardware as in the main experiments. The 50-step schedule is retained except for the

explicitly identified reduced-step baseline. For methods with multiple thresholds or operating modes, operating points are selected to target the reported speedup ranges; the exact configurations are listed in Table 1.

B.2 Branch-wise Proxy Definitions Used by RA-CFGCache

RA-CFGCache does not introduce a new branch-local error estimator. Instead, it takes the control signal of a base caching method as the branch-wise proxy in Eq. 11. For each branch $j \in \{u, c\}$, we denote this signal by $\hat{r}_j(t_a, t_b)$. The proxy is evaluated independently for the two CFG branches before guided-risk composition. Importantly, the proxy families used in our experiments differ in whether they depend on the current sample: Tea-style and DiCache-style proxies use online feature signals, whereas the MagCache-style proxy is derived from fixed offline-calibrated magnitude-ratio statistics.

MagCache-style proxy. For the FLUX.1-dev main experiments, we use the fixed magnitude-ratio statistics of MagCache [Ma et al., 2025]. Let $\mu_j(k)$ denote the pre-calibrated magnitude ratio associated with branch j at execution step k. For a candidate reuse interval from anchor a to current step b, we form the cumulative ratio

$$M_j(a, b) = \prod_{k=a+1}^{b} \mu_j(k), \tag{16}$$

and use the corresponding deviation

$$\hat{r}_j(t_a, t_b) = |1 - M_j(a, b)| \tag{17}$$

as the branch-wise proxy supplied to the CFG-aware composition. The magnitude-ratio sequence is fixed before evaluation and no sample-specific feature measurement is used to construct this proxy. Thus, under a fixed model, timestep schedule, calibration table, and threshold, the MagCache-style proxy should not be interpreted as an instance-adaptive estimator.

TeaCache-style proxy. Following the core design of TeaCache [Liu et al., 2025a], we construct a sample-dependent control signal from the relative change of the timestep-modulated feature. For each CFG branch j, let $m_j(k)$ denote the current modulated feature and m_j^{ref} the stored reference feature maintained by the corresponding TeaCache-compatible implementation. We compute

$$d_j(k) = \frac{\mathrm{mean}\left|m_j(k) - m_j^{\mathrm{ref}}\right|}{\mathrm{mean}\left|m_j^{\mathrm{ref}}\right| + 10^{-6}}. \tag{18}$$

If the reference mean absolute magnitude is zero, the raw change signal $d_j(k)$ is set to zero before rescaling. We then apply the same model-specific polynomial p_{model} as the corresponding TeaCache-compatible baseline and define

$$\hat{r}_j(k) = \max\{p_{\mathrm{model}}(d_j(k)),\, 0\}\,, \tag{19}$$

with no upper clipping. Thus, the proxy preserves TeaCache's basic relative-change and polynomial-rescaling construction while serving as the branch-wise online signal for **RA-CFGCache**. The resulting proxy is sample-dependent; **RA-CFGCache** replaces the original TeaCache decision rule with the guided-risk composition and cumulative controller described in Section 3.4.

DiCache-style proxy. For the DiCache-style compatibility experiments, we use the online shallow-probe signal of DiCache [Bu et al., 2025]. For each branch j, the first m Transformer blocks are evaluated and the relative ℓ_1 change of the resulting shallow feature is used as the branch-wise proxy:

$$\hat{r}_j(k) = \frac{\|z_j^{(m)}(k) - z_j^{(m)}(k-1)\|_1}{\|z_j^{(m)}(k-1)\|_1 + \epsilon}, \tag{20}$$

where $z_j^{(m)}(k)$ denotes the feature after the probe depth m. We use $m = 2$ in the reported experiments. Because the probe feature is computed from the current branch trajectory, this proxy is sample-dependent. The probe computation is included in the reported end-to-end latency.

Normalization and accumulation. No additional sample-wise normalization is introduced by **RA-CFGCache** beyond the normalization or calibration already used by the corresponding base proxy. The two branch scores are composed through Eq. 11; propagation rescaling is then applied as in Eq. 13. The interval-level accumulation variable R in Eq. 14 belongs to **RA-CFGCache** and is separate from the definition of the branch-wise proxy itself.

For the Tea-style and DiCache-style estimators, the base proxy is an adjacent-step online change signal rather than a direct estimate of the anchor-to-target prediction error. **RA-CFGCache** uses this signal as the current branch-wise contribution within the active reuse interval, while the cross-branch alignment statistic is indexed by the current cache anchor and target step.

B.3 Calibration Details

The offline calibration procedure estimates two quantities used by **RA-CFGCache**: the anchor–target alignment statistic $\bar{\rho}_{t_a,t_b}$ and the timestep-dependent propagation gain g_t. The resulting calibration quantities are fixed during evaluation.

Calibration of $\bar{\rho}$. We first run full, non-accelerated inference and record the hidden-space residual cached by the corresponding implementation at each timestep. Let $h^c_{t_k}$ and $h^u_{t_k}$ denote the conditional and unconditional residual tensors at sampler step k, whose diffusion time is t_k. These hidden-space quantities are distinct from the final branch predictions c_{t_k} and u_{t_k} used in the CFG error formulation.

For FLUX.1-dev, the cached residual is the change in image-token hidden states across the complete double-stream and single-stream Transformer stack: the hidden states immediately before `final_layer` minus the image hidden states after `img_in` and before the block stack. Only the image-stream residual is cached. For Wan2.1-T2V-1.3B, it is the change in video hidden states across the complete Transformer stack, before `norm_out`, modulation, and `proj_out`. For CogVideoX-2B, the corresponding video-token and text-token residuals across the full Transformer stack are cached separately before the final normalization and projection layers.

For anchor step a and target step $b > a$, we construct

$$d_c(t_a, t_b) = \mathrm{vec}\big(h^c_{t_a} - h^c_{t_b}\big), \qquad d_u(t_a, t_b) = \mathrm{vec}\big(h^u_{t_a} - h^u_{t_b}\big), \tag{21}$$

where $\mathrm{vec}(\cdot)$ flattens the tensor into a vector. We average their cosine similarity over calibration samples to obtain $\bar{\rho}_{t_a,t_b}$. Indexed by execution order, $\bar{\rho}$ is an upper-triangular anchor–target matrix. It describes cross-branch alignment of hidden-residual drift and serves as a calibrated proxy for the prediction-error alignment appearing in the exact CFG error identity. It is not itself the actual prediction-error alignment for every online reuse event.

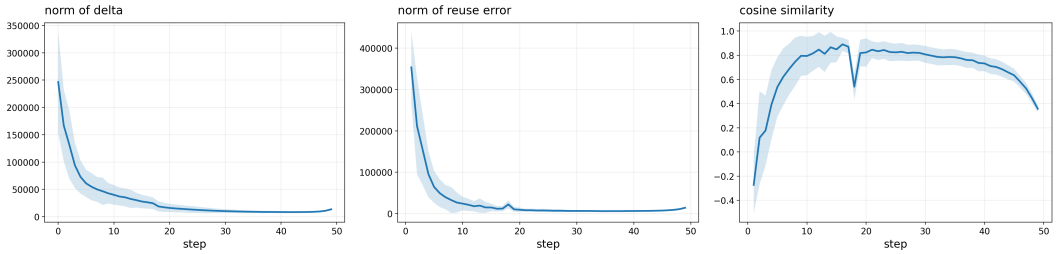
Calibration of propagation-aware rescaling. Propagation calibration uses an isolated one-step prediction perturbation. At a selected execution step, both the conditional and unconditional branch predictions used by the sampler are replaced by their corresponding predictions from the preceding execution step. The reused predictions are then combined using the same CFG rule as in normal inference. All other steps use fresh computation on their current latent states, so each perturbed trajectory contains only one injected reuse event.

The perturbed and reference trajectories use the same prompt, initial noise, and sampling randomness. We measure the discrepancy between the perturbed and fresh guided predictions at the injected step and the resulting deviation between their final latents before VAE decoding. For each calibration sample, we form the ratio between final and local discrepancies, and then average these per-sample ratios at each probed step. Samples with invalid or vanishing local discrepancy are excluded from the ratio average.

We probe one eligible step every $\Delta_g = 3$ execution steps, excluding the first denoising step because no preceding prediction is available. The resulting timestep-dependent mean gains are fitted using the power-law surrogate in Eq. 12 by least squares. Fitted gains are lower-bounded at zero when used for risk rescaling.

Offline calibration cost and online overhead. Let C denote the number of calibration prompts, T the number of denoising steps, F_m the cost of one full two-branch denoising evaluation, and D_m the size of a saved hidden residual tensor. Alignment calibration consists of full-trajectory collection followed by pairwise reduction, with complexity

$$\mathcal{O}(CTF_m) + \mathcal{O}(CT^2 D_m).$$



(a) $\|\Delta_t\|_1$ (b) $\|\Delta_t - \Delta_{t-1}\|_1$ (c) $\cos(\Delta_t, \Delta_{t-1})$

Figure 7: **Temporal dynamics of the conditional correction signal under CFG.** From left to right: the norm of Δ_t, the adjacent-step variation $\|\Delta_t - \Delta_{t-1}\|_1$, and the cosine similarity $\cos(\Delta_t, \Delta_{t-1})$ between consecutive correction vectors.

The quadratic term operates only on stored residual tensors and does not require $\mathcal{O}(T^2)$ diffusion-model evaluations. Propagation calibration probes one perturbation anchor every Δ_g steps, giving an approximate rollout cost

$$\mathcal{O}(CT^2F_m/\Delta_g).$$

For a 100-prompt FLUX.1-dev calibration configuration with $T = 50$, alignment and propagation calibration require approximately 0.60 and 8.3 GPU-hours, respectively, for 8.9 GPU-hours in total. Raw calibration traces temporarily occupy approximately 11 GB at 1024×1024 resolution and 42 GB at 2048×2048, whereas the final calibrated statistics occupy only tens of kilobytes. Calibration is entirely offline.

At test time, the **RA-CFGCache** control layer adds only constant-time lookup and scalar composition/rescaling beyond the computations required by the base proxy, and introduces no additional diffusion-model forward pass. This online overhead is included in the reported end-to-end latency.

Calibration reuse and transfer. Once calibrated, $\bar{\rho}_{t_a,t_b}$ and g_t are fixed and reused across evaluation prompts, random seeds, and operating thresholds τ. The alignment statistic is calibrated separately for the evaluated CFG scales, while the same propagation prior is reused across those scale settings. Changes to the model or sampler/noise schedule require recalibration; changes to the timestep grid or prompt domain should be validated on the new setting and recalibrated when necessary.

Separate calibration-size ablations show stable performance when reducing alignment calibration to 50 prompts or propagation calibration to 10 prompts in the evaluated setting. These individual ablations do not establish the performance of a jointly reduced 50/10-prompt configuration.

C Mechanism Analysis

C.1 CFG Error Geometry under Classifier-Free Guidance

Branch-Error Alignment and Conditional-Correction Drift.

We examine how the evolution of the conditional correction relates to branch-error alignment. Let

$$\Delta_t = c_t - u_t, \qquad \tilde{\Delta}_t = \tilde{c}_t - \tilde{u}_t. \tag{22}$$

Here Δ_t is the difference between the fresh conditional and unconditional final branch predictions. For arbitrary branch approximations, the following identity holds:

$$\delta c_t = \delta u_t + (\tilde{\Delta}_t - \Delta_t). \tag{23}$$

Thus, for nonzero branch errors,

$$\rho_t = \cos\Big(\delta u_t,\ \delta u_t + (\tilde{\Delta}_t - \Delta_t)\Big). \tag{24}$$

For direct prediction reuse from anchor t_a, $\tilde{\Delta}_t = \Delta_{t_a}$. In particular, adjacent-step prediction reuse gives the correction drift $\Delta_{t_{k-1}} - \Delta_{t_k}$ at execution step k. For intermediate-feature or residual caching, this temporal-drift interpretation serves as a proxy rather than an exact identity for the induced prediction error.

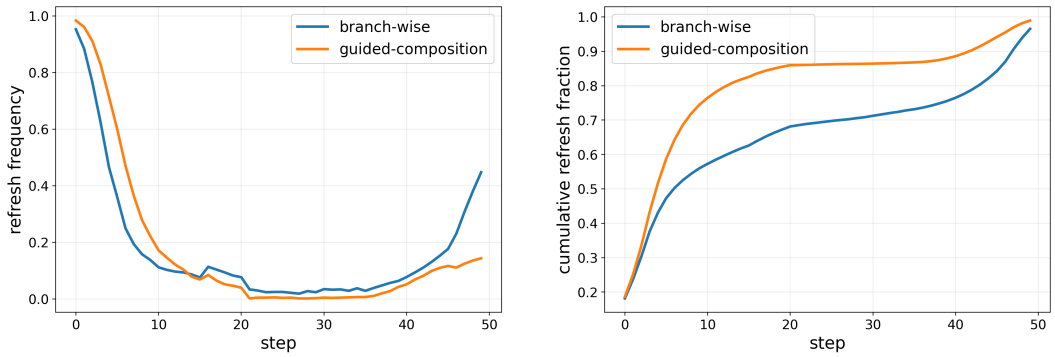


Figure 8: **Refresh redistribution under matched budget.** Left: per-step refresh frequency of the branch-wise and guided-composition controllers. Right: cumulative refresh fraction over timesteps. Guided-composition shifts refresh decisions toward earlier timesteps, showing that preserving CFG error geometry changes the actual scheduling behavior rather than only the local risk value.

Alignment depends on both the magnitude and direction of the correction error relative to δu_t. If the correction error is small relative to a nonzero unconditional error, the two branch errors remain closely aligned. A larger correction error may preserve or weaken alignment depending on its direction. Neither the correction norm alone nor its unnormalized temporal variation determines ρ_t.

Figure 7 examines the correction norm, adjacent-step variation, and adjacent-step cosine similarity using the final branch predictions $\Delta_t = c_t - u_t$. In the displayed setting, the correction norm decreases over sampling, while its temporal variation is large early, smaller in the middle, and rises again near the end. These observations describe changing correction dynamics and are consistent with timestep-dependent branch alignment. They do not, by themselves, establish that correction-norm decay causes increased alignment; the relative drift magnitudes and directions also matter.

Risk Mismatch and Refresh Redistribution under CFG.

The calibrated alignment term changes the temporal profile of the proxy score. We examine whether this change also affects refresh allocation through an offline replay on precomputed branch traces. The replay isolates scheduling behavior and does not measure end-to-end generation quality.

We compare two cumulative-threshold controllers using the same base proxy, reuse action, and replay protocol: a no-cross-term controller and a guided-composition controller that includes the calibrated cross-term. Their thresholds are tuned separately to match the average refresh count.

Figure 8 shows that guided composition shifts refreshes toward earlier timesteps at the matched average count. The mean refresh step decreases from 16.87 to 10.03, while the fraction of refreshes assigned to the first 20% of steps increases from 56.1% to 75.0%. These results show that the calibrated cross-term changes the allocation of refresh decisions in the replay setting. Whether this redistribution improves output fidelity is evaluated separately through controlled quality comparisons.

C.2 Propagation-aware Rescaling under Diffusion Dynamics

The guided error describes a local prediction perturbation, whereas its effect on the final latent also depends on the sampler and the remaining denoising dynamics. We use a first-order analysis to motivate the empirical propagation prior.

First-order propagation analysis. To distinguish execution order from diffusion time, let z_k denote the latent entering sampler step k, with corresponding diffusion time t_k. Write one complete update as

$$z_{k+1} = F_k(z_k, p_{t_k}(z_k)) .$$

Let e_k be a guided-prediction perturbation introduced only at step k. Evaluating derivatives along the unperturbed trajectory gives

$$\delta z_{k+1} \approx B_k e_k, \qquad B_k := \frac{\partial F_k}{\partial p},$$

and define the effective Jacobian of a subsequent complete step as

$$A_j := \frac{d}{dz_j} F_j\big(z_j, p_{t_j}(z_j)\big) = \frac{\partial F_j}{\partial z_j} + \frac{\partial F_j}{\partial p}\frac{\partial p_{t_j}}{\partial z_j}.$$

The final perturbation is therefore approximated by the ordered product

$$\delta z_{T+1}^{(k)} \approx J_k e_k, \qquad J_k := A_T A_{T-1} \cdots A_{k+1} B_k,$$

where the product preceding B_k is the identity for the final update. For multistep samplers, the state can be augmented to include the history required by the update. For stochastic samplers, the comparison conditions on shared sampling randomness.

For a nonzero perturbation, the directional amplification factor is

$$g_{t_k}(e_k) := \frac{\|\delta z_{T+1}^{(k)}\|_2}{\|e_k\|_2} \approx \frac{\|J_k e_k\|_2}{\|e_k\|_2}.$$

This factor depends on both the trajectory and the perturbation direction. We summarize this behavior using a timestep-dependent calibration average,

$$\bar{g}_{t_k} := \mathbb{E}_{\text{cal}}\left[\frac{\|\delta z_{T+1}^{(k)}\|_2}{\|e_k\|_2}\right].$$

In practice, we do not explicitly compute the Jacobian chain. Instead, the calibration uses complete perturbed rollouts and measures the downstream effect of an isolated one-step perturbation. The fitted $\hat{g}_{t_k}$ is therefore an empirical timestep-dependent surrogate motivated by the first-order analysis rather than a direct Jacobian estimate.

From single-step sensitivity to cache control. Under repeated reuse, earlier errors change the states at which later model evaluations occur, and perturbations may interact. The single-step analysis therefore does not provide an exact decomposition of final error under arbitrary multi-step caching.

We use the fitted gain as a lightweight timestep-dependent prior. Given the guided-risk contribution $\hat{r}^{\text{guided}}_{t_a,t_b}$ for a candidate reuse decision from anchor t_a to target t_b, we define

$$\Delta\hat{r}_{t_a,t_b} = \max\left(0, \hat{g}_{t_b}\hat{r}^{\text{guided}}_{t_a,t_b}\right).$$

When reuse is accepted, the accumulated control score is updated as

$$R \leftarrow R + \Delta\hat{r}_{t_a,t_b}.$$

Thus, propagation weighting is applied before accumulation. The resulting R is an online control statistic, not a bound on the final output deviation.

C.3 Reuse Object Selection: Why Joint Branch Reuse Is a Strong Default

Beyond refresh scheduling, CFG caching introduces a second decision dimension: what should be reused between two refresh steps. Under true CFG,

$$p_t = u_t + s\Delta_t, \qquad \Delta_t := c_t - u_t,$$

or equivalently,

$$p_t = (1-s)u_t + sc_t.$$

For direct prediction reuse from anchor t_a, different actions induce the following exact guided errors:

$$\delta p_t^{(u)} = (1-s)(u_{t_a} - u_t),$$

$$\delta p_t^{(c)} = s(c_{t_a} - c_t),$$

$$\delta p_t^{(\Delta)} = s(\Delta_{t_a} - \Delta_t),$$

and

$$\delta p_t^{(\text{both})} = (1-s)(u_{t_a} - u_t) + s(c_{t_a} - c_t).$$

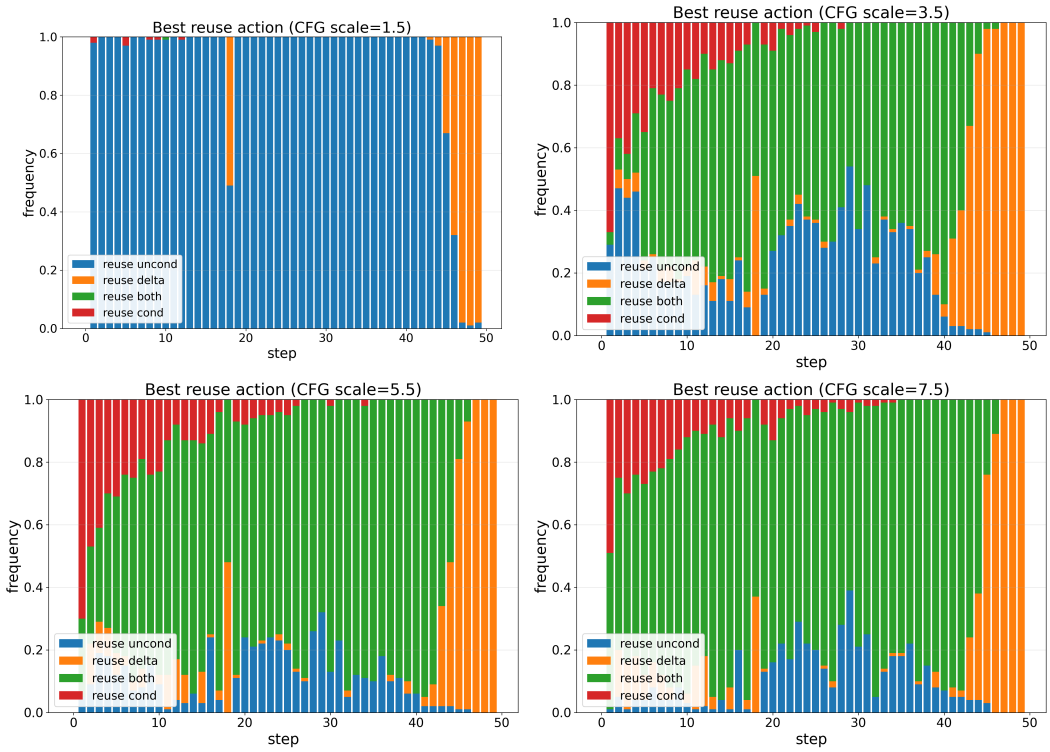


Figure 9: **Best reuse action frequency across timesteps under different CFG scales** (top-left: 1.5, top-right: 3.5, bottom-left: 5.5, bottom-right: 7.5). As CFG scale increases, reuse-both dominates a larger portion of timesteps, while reuse-Δ becomes favorable mainly in the very late stage. Reuse-cond is rarely optimal throughout.

For intermediate-feature or residual caching, these equations describe the corresponding idealized prediction-reuse actions rather than the exact internal computation.

In the direct-prediction analysis, reuse-Δ recomputes the unconditional prediction and reuses the cached CFG delta, saving one branch evaluation. Reuse-uncond and reuse-cond also save one branch evaluation, whereas reuse-both saves two. For intermediate caching, the exact saving depends on the cached components and any computations retained for proxy evaluation.

We first examine object selection using a timestep-wise offline action oracle based on true guided error. For each CFG scale, we enumerate candidate reuse actions at every timestep and record, over 100 prompts, how often each action attains the smallest guided action error relative to full computation. As shown in Fig. 9, the preferred reuse object is phase-dependent. At low CFG scale, reuse-uncond remains competitive over a larger portion of the trajectory. As CFG scale increases, reuse-both is selected more frequently, while reuse-Δ becomes competitive mainly in the late stage. Reuse-cond is rarely preferred.

The late-stage competitiveness of reuse-Δ is consistent with the CFG geometry. The CFG-delta drift is

$$\Delta_{t_a} - \Delta_t = (c_{t_a} - c_t) - (u_{t_a} - u_t).$$

When the conditional and unconditional branch drifts are both highly aligned and similar in magnitude, this difference can become small. In that regime, directly reusing the CFG delta can induce a relatively small raw guided error.

A default online reuse action should also account for computational saving. We therefore additionally compare actions by guided error per saved branch evaluation. Under this diagnostic ratio, reuse-both performs best overall across the evaluated CFG scales in our offline analysis. This ratio is an auxiliary diagnostic and does not establish an optimal online policy.

Taken together, object choice under CFG is phase-dependent, while joint branch reuse provides a strong practical default in the evaluated settings. It is frequently preferred at moderate-to-large CFG

Table 9: **Comparison with CFG-aware guidance and acceleration methods on FLUX.1-dev. RA-CFGCache** preserves the original sampler and fixed CFG rule. MAMBO-G, CFG Scheduler, and OUSAC modify the guidance policy, while FasterCache exploits CFG branch redundancy. LPIPS, SSIM, and PSNR measure sampler fidelity, with CLIPScore, ImageReward, and HPSv2 reported as complementary quality metrics.

Method	Speedup↑	LPIPS↓	SSIM↑	PSNR↑	CLIP↑	ImageReward↑	HPSv2↑
Vanilla ($T = 50$)	1.00×	0.0000	1.0000	∞	27.9063	0.8003	0.2776
Vanilla ($T = 25$)	1.96×	0.4556	0.6207	13.09	27.0429	0.2629	0.2364
FasterCache	2.09×	0.2158	0.7593	20.88	27.4779	0.6241	0.2643
MAMBO-G (T=25)	1.98×	0.5616	0.5192	11.64	**28.4720**	0.9800	0.2930
CFG Scheduler	1.00×	0.5327	0.5503	12.26	28.1390	**1.0160**	**0.3000**
OUSAC	3.44×	0.5530	0.5092	11.74	27.8770	0.9300	0.2880
RA-CFGCache-Slow	3.19×	**0.1137**	**0.8680**	**24.02**	27.8843	0.7672	0.2748
RA-CFGCache-Fast	**3.95**×	0.1484	0.8340	22.75	27.9071	0.7403	0.2694

scales and provides favorable error per saved branch evaluation. We therefore use joint branch reuse as the default action in the main controller.

D Extended Experimental Results

D.1 Comparison with CFG-aware Guidance and Acceleration Methods

We further compare **RA-CFGCache** with related CFG-aware guidance and acceleration methods, including MAMBO-G Zhu et al. [2026], CFG Scheduler Wang et al. [2024], and OUSAC Sun et al. [2025], under the same FLUX.1-dev evaluation protocol.

Importantly, **RA-CFGCache does not tune the CFG scale or modify the guidance schedule**. It preserves the original 50-step sampler and fixed CFG rule throughout inference and only controls whether intermediate computation is refreshed or reused. Its objective is therefore to accelerate a fixed vanilla CFG sampler while preserving its outputs as closely as possible, rather than to optimize absolute generation quality or preference-oriented metrics.

In contrast, MAMBO-G dynamically adjusts guidance magnitudes, CFG Scheduler varies guidance weights across timesteps, and OUSAC jointly optimizes sparse guidance scheduling and caching. These approaches may therefore alter the target generation trajectory itself, whereas **RA-CFGCache** treats the vanilla CFG sampler as fixed.

At the closest high-speed operating point, **RA-CFGCache-Fast** achieves a $3.95\times$ speedup compared with $3.44\times$ for OUSAC, while producing outputs substantially closer to the vanilla outputs. Specifically, LPIPS decreases from 0.5530 to 0.1484, SSIM increases from 0.5092 to 0.8340, and PSNR increases from 11.74 to 22.75. The corresponding CLIPScore remains comparable (27.9071 vs. 27.8770). **RA-CFGCache-Slow** provides a more fidelity-oriented operating point, achieving LPIPS 0.1137, SSIM 0.8680, and PSNR 24.02 at a $3.19\times$ speedup.

MAMBO-G, CFG Scheduler, and OUSAC obtain higher ImageReward or HPSv2 scores in several settings. This distinction is consistent with their different objectives: these methods explicitly modify the guidance policy, whereas **RA-CFGCache** does not optimize CFG parameters or preference-oriented generation quality. Our claim is therefore a stronger *speed–sampler-fidelity trade-off*, rather than universal superiority in absolute generation quality.

The two directions are therefore complementary: guidance-policy optimization changes the target trajectory to improve generation characteristics, whereas **RA-CFGCache** accelerates a fixed guidance policy and aims to reproduce its outputs as faithfully as possible.

D.2 Generality of Propagation-aware Rescaling

Although propagation-aware rescaling is introduced together with our CFG-aware guided-risk formulation, its role is more general: it corrects local reuse scores by accounting for the non-uniform downstream impact of errors across timesteps. We therefore evaluate it in a standard single-prediction caching setting, where reuse decisions are made on a single model prediction rather than CFG-composed branch outputs.

Table 10: **Generality of propagation-aware rescaling in the standard single-prediction setting.** We apply propagation-aware rescaling to several representative non-CFG caching baselines and observe consistent fidelity improvements across the evaluated proxy families.

Base method	Variant	Speedup↑	LPIPS↓	SSIM↑	PSNR↑
TeaCache	Original	3.27×	0.3617	0.7112	17.05
TeaCache	+ Propagation Rescaling	**3.32**×	**0.3207**	**0.7305**	**18.02**
MagCache	Original	3.11×	0.1964	0.8169	22.05
MagCache	+ Propagation Rescaling	**3.12**×	**0.1822**	**0.8266**	**22.57**
DiCache	Original	**3.20**×	0.2999	0.7744	21.58
DiCache	+ Propagation Rescaling	3.19×	**0.2393**	**0.7966**	**22.38**

Table 11: **Compatibility with different base proxy estimators.** We apply **RA-CFGCache** on top of different branch-wise proxy families and compare it against their original branch-wise accumulation controllers. Across all proxy types, **RA-CFGCache** consistently improves fidelity at nearly unchanged latency, showing that its gain is complementary to the underlying proxy design rather than tied to a specific estimator.

Base proxy family	Controller	Latency↓	Speedup↑	LPIPS↓	SSIM↑	PSNR↑
Tea-style proxy	Branch-wise accumulation	6.16 s	3.41×	0.1852	0.7977	20.60
	+ RA-CFGCache wrapper	**6.16 s**	**3.41**×	**0.1492**	**0.8297**	**22.39**
Mag-style proxy	Branch-wise accumulation	6.18 s	3.40×	0.1683	0.8144	21.35
	+ RA-CFGCache wrapper	**6.17 s**	**3.41**×	**0.1232**	**0.8590**	**23.53**
DiCache-style proxy	Branch-wise accumulation	6.13 s	3.43×	0.2177	0.7706	19.48
	+ RA-CFGCache wrapper	**6.13 s**	**3.43**×	**0.1453**	**0.8364**	**22.57**

Specifically, we apply the same propagation-aware rescaling to representative proxy baselines, including TeaCache-style, MagCache-style, and DiCache-style controllers. We keep their original proxy signals, thresholds, and reuse policies unchanged, and modify only the local reuse score by incorporating the timestep-dependent propagation coefficient.

Table 10 shows that propagation-aware rescaling consistently improves the final efficiency–fidelity trade-off across all tested baselines. In particular, it reduces LPIPS and improves SSIM and PSNR while maintaining nearly identical speedup.

These results indicate that propagation-aware rescaling is not merely an implementation detail tied to **RA-CFGCache**, but a broadly applicable correction principle: local reuse error should be adjusted according to its timestep-dependent downstream propagation before being used for decision-making. This supports our formulation that effective cache control requires not only accurate local error estimation, but also a proper modeling of how such errors propagate to the final output.

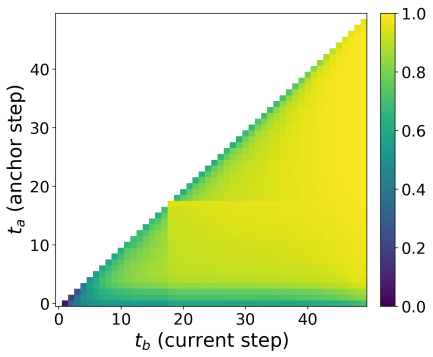
D.3 Compatibility with Different Base Proxy Estimators

RA-CFGCache is designed as a control framework that operates on top of branch-wise proxy signals, rather than relying on a specific proxy design. To verify this plug-in property, we evaluate our method with several representative proxy families, including Tea-style, Mag-style, and DiCache-style estimators.

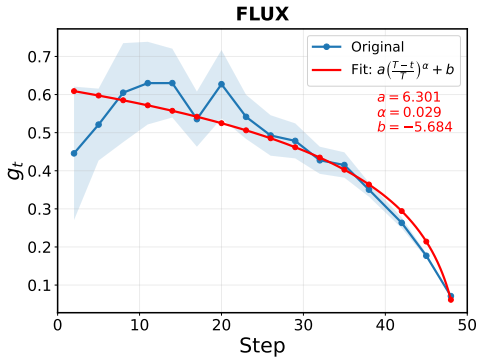
For each proxy family, we compare its original branch-wise accumulation controller with the corresponding **RA-CFGCache** version, which replaces the original accumulation rule with CFG-aware guided-risk composition and propagation-aware rescaling, while keeping the underlying proxy signals unchanged.

As shown in Table 11, **RA-CFGCache** consistently improves fidelity across all proxy families at nearly identical latency. For example, under the Tea-style proxy, LPIPS is significantly reduced with improved SSIM and PSNR, and similar improvements are observed for Mag-style and DiCache-style proxies.

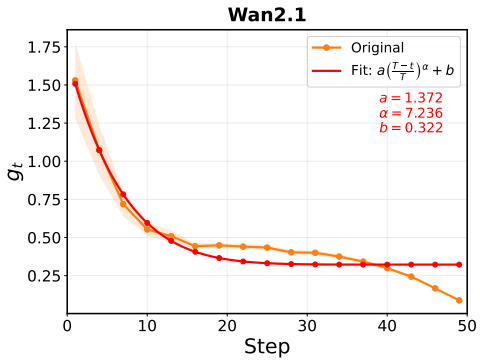
These results demonstrate that the gain of **RA-CFGCache** does not come from designing a stronger proxy signal, but from correcting how proxy signals are composed and accumulated under CFG. This



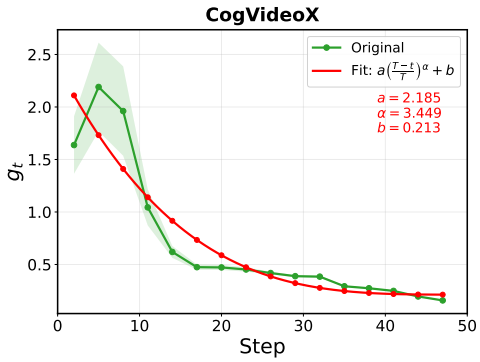
(a) Pairwise alignment heatmap $\bar{\rho}_{t_a,t_b}$.



(b) Power-law fit of the propagation gain curve g_t on FLUX.



(c) Power-law fit of the propagation gain curve g_t on Wan2.1-T2V-1.3B.



(d) Power-law fit of the propagation gain curve g_t on CogVideoX.

Figure 10: **Additional calibration visualizations.** Top-left: pairwise branch-error alignment heatmap. The other three panels show model-specific power-law fits of the propagation gain curve g_t.

confirms that **RA-CFGCache** can be viewed as a general wrapper that upgrades existing proxy-based caching methods into a risk-aligned control paradigm.

D.4 Additional Calibration Visualizations

We visualize the two calibrated quantities used by our method: the anchor–target alignment statistic $\bar{\rho}_{t_a,t_b}$ and the timestep-dependent propagation gain g_t.

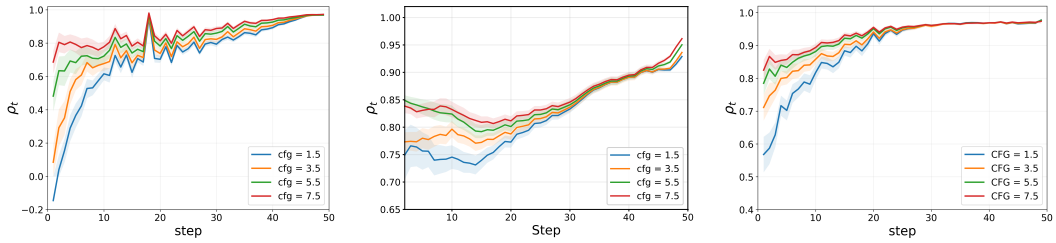
Figure 10(a) shows the alignment heatmap between anchor step t_a and current step t_b. The heatmap exhibits clear structure rather than an unstructured noisy pattern. Alignment is weaker and more variable in some early regions and stronger over broad later regions. This motivates retaining anchor–target dependence in the calibrated alignment statistic, while the heatmap alone does not establish that a full pairwise matrix is necessary relative to simpler parameterizations.

Figure 10(b)–(d) show the measured propagation-gain curve and its fitted power-law surrogate. The measured gain is non-uniform across timesteps: perturbations introduced at different steps lead to substantially different final deviations. The fitted curve captures the dominant trend and provides a compact representation for online control. Its usefulness for cache control is evaluated separately through the surrogate ablation.

Together, these visualizations illustrate the two calibration objects used by **RA-CFGCache**: an anchor–target calibrated alignment statistic for guided-risk composition and timestep-dependent propagation weighting for control-score accumulation.

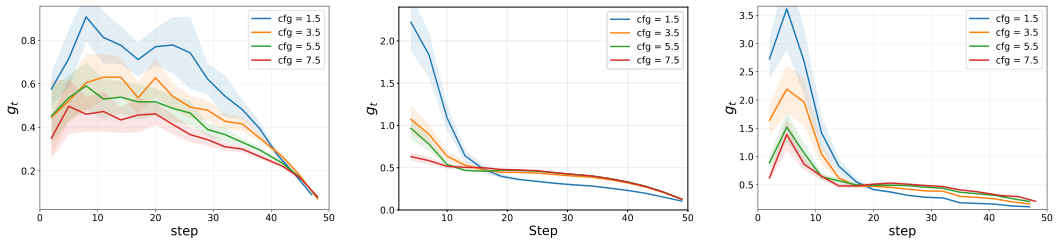
D.5 Cross-model Alignment and Propagation Patterns

To further examine the empirical patterns relevant to our calibration procedure, we visualize alignment and propagation statistics across diffusion models and CFG settings.



(a) FLUX (b) Wan2.1 (c) CogVideoX

Figure 11: **Alignment curves across CFG scales for different diffusion models.** Across models, ρ_t generally remains lower or less stable in the early-to-middle stage and tends toward a higher-alignment regime in later timesteps, although detailed local fluctuations are model-dependent.



(a) FLUX (b) Wan2.1 (c) CogVideoX

Figure 12: **Propagation gain curves g_t across CFG scales for different diffusion models.** The propagation gain is strongly timestep-dependent, with larger values in the early or early-middle stages and much smaller values in later steps. Across CFG scales, the curves show model-dependent magnitude differences and tend to become closer in later steps, indicating that local guided errors have substantially different final impacts depending on their injection timestep.

Figure 11 shows the timestep-wise alignment curves across multiple CFG scales for the evaluated models. Despite differences in architecture and absolute magnitude, the curves exhibit structured timestep-dependent variation, with many settings moving toward a stronger alignment regime later in sampling.

Figure 12 presents the corresponding propagation-gain curves. Across the evaluated models, perturbations introduced at different timesteps have markedly different downstream effects, with the dominant trend generally decreasing toward later sampling stages.

These results show related empirical patterns across the evaluated models and CFG configurations. They are consistent with our use of model-specific alignment calibration together with a shared low-dimensional functional family for the propagation prior. However, similar qualitative patterns do not imply direct transfer of calibrated parameters across models, samplers, or guidance configurations.

D.6 Why a Power-law Surrogate for Propagation Gain

After measuring the propagation gain through offline perturbation calibration, we require a practical surrogate for online scheduling. Our goal is not to reproduce every local fluctuation of the measured curve, but to obtain a smooth control signal that captures the dominant timestep dependence.

Using the increasing execution-step index k, we model the gain as

$$\hat{g}_k = \lambda\Big(\frac{T-k}{T}\Big)^{\alpha} + \beta,$$

where t_k denotes the corresponding diffusion time and λ, α, β are fitted from the offline perturbation measurements.

The power-law form has three practical advantages. First, it provides a smooth approximation to the dominant propagation trend and avoids reacting to local calibration noise. Second, $(T-k)/T$ gives a dimensionless parameterization of sampling progress, although this normalization alone does not guarantee transfer across step counts or noise schedules. Third, the surrogate requires only three scalar parameters and therefore adds negligible online cost.

Table 12: **Effect of propagation-gain surrogate.** We compare compact surrogate forms for propagation-aware rescaling. The power-law surrogate provides the best fidelity among the tested smooth parameterizations, achieving the lowest LPIPS and highest SSIM with competitive PSNR.

Surrogate	$R^2 \uparrow$	**NMAE** $\downarrow$	**Speedup** $\uparrow$	**LPIPS** $\downarrow$	**SSIM** $\uparrow$	**PSNR** $\uparrow$
No prop. rescaling	–	–	3.74×	0.1699	0.8162	21.41
Fixed linear decay	-0.0024	0.3174	3.75×	0.1469	0.8297	22.74
Fitted linear decay	0.7164	0.1521	3.76×	0.1448	0.8379	22.80
Power-law decay	**0.8201**	**0.1010**	**3.77**×	**0.1406**	**0.8398**	**22.93**

Table 13: **Effect of caching strategies under CFG parallelism.** Controlled FLUX.1-dev study at 1024×1024 resolution, CFG scale 3.5, 50 denoising steps, comparing independent branch-wise caching and joint guided caching under a matched branch-evaluation budget.

Controller	**Reuse both (%)** $\uparrow$	**One-sided reuse (%)** $\downarrow$	Full-step refresh (%)	**Latency (s)** $\downarrow$
Branch-wise caching	35.1	53.8	11.1	6.81
Ours (reuse both)	**62.0**	**0.0**	38.0	**4.21**

Figure 12 shows that the measured gains contain model-dependent transients and local fluctuations but exhibit a clear coarse timestep dependence. Table 12 compares several propagation-gain surrogates. Removing propagation rescaling gives the weakest final fidelity in the reported comparison, supporting the benefit of timestep-dependent downstream weighting. Compared with the fixed and fitted linear alternatives, the power-law surrogate achieves the best LPIPS and SSIM with competitive PSNR at comparable speedup. The R^2 and NMAE columns characterize approximation quality, whereas LPIPS, SSIM, and PSNR evaluate the downstream effect of the resulting controller.

The power-law form should therefore be understood as a practical scheduling prior rather than a strict law of diffusion error propagation.

D.7 Parallel Consistency under CFG Parallelism

We separately evaluate a parallel CFG implementation, distinct from the single-GPU sequential branch execution used in the main experiments. Each sample uses two GPUs, with the conditional and unconditional branches placed on separate GPUs. Their outputs are synchronized before CFG composition, and the associated synchronization overhead is included in the reported end-to-end latency.

We compare joint branch decisions against independent TeaCache-style controllers applied separately to the conditional and unconditional branches. The thresholds are tuned to match the aggregate number of branch evaluations.

As shown in Table 13, independent branch-wise control uses one-sided reuse at 53.8% of steps and joint reuse at 35.1%. One-sided reuse reduces branch computation, but may provide limited wall-clock benefit when the recomputed branch remains on the critical path. Joint control avoids these one-sided decisions and skips recomputation of cached components in both branches at each accepted joint reuse step.

At the matched branch-evaluation count, independent branch-wise control requires 6.81 s, whereas joint control requires 4.21 s. These results support joint reuse as an effective execution choice in the evaluated parallel implementation. They do not imply that one-sided reuse provides no computational saving or that the same latency advantage necessarily holds across different hardware configurations or execution backends.

E Broader impacts

RA-CFGCache improves the efficiency of CFG-based diffusion inference, which can lower deployment cost and reduce energy consumption for visual generation systems. The method does not introduce new generative capabilities, datasets, or pretrained models, and we do not identify negative societal impacts unique to the proposed caching algorithm. As with other efficiency improvements for generative models, practical deployment should still follow the safety and content-moderation policies of the underlying models.

F Additional Qualitative Results

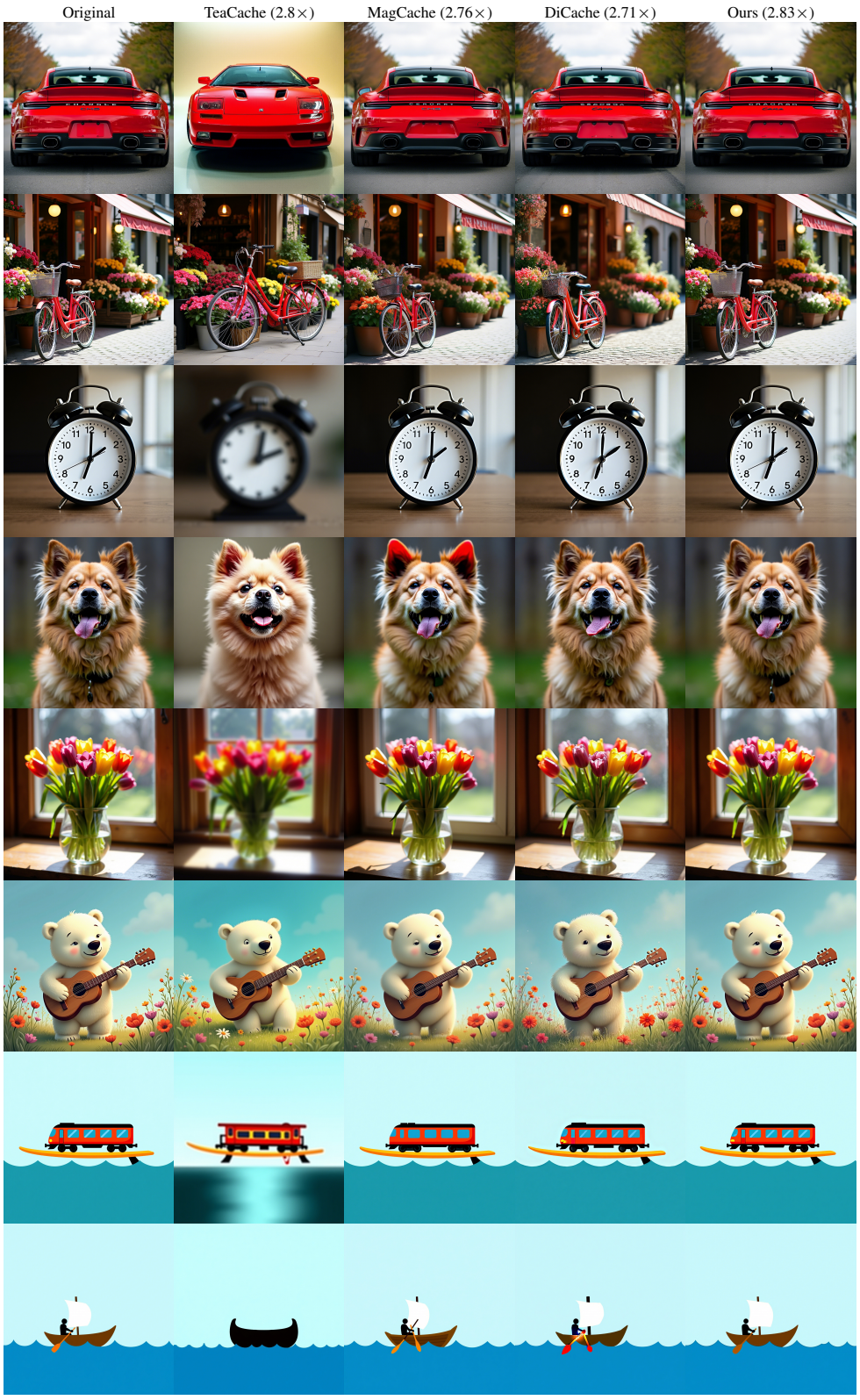


Figure 13: **Qualitative comparison on FLUX.1-dev.**

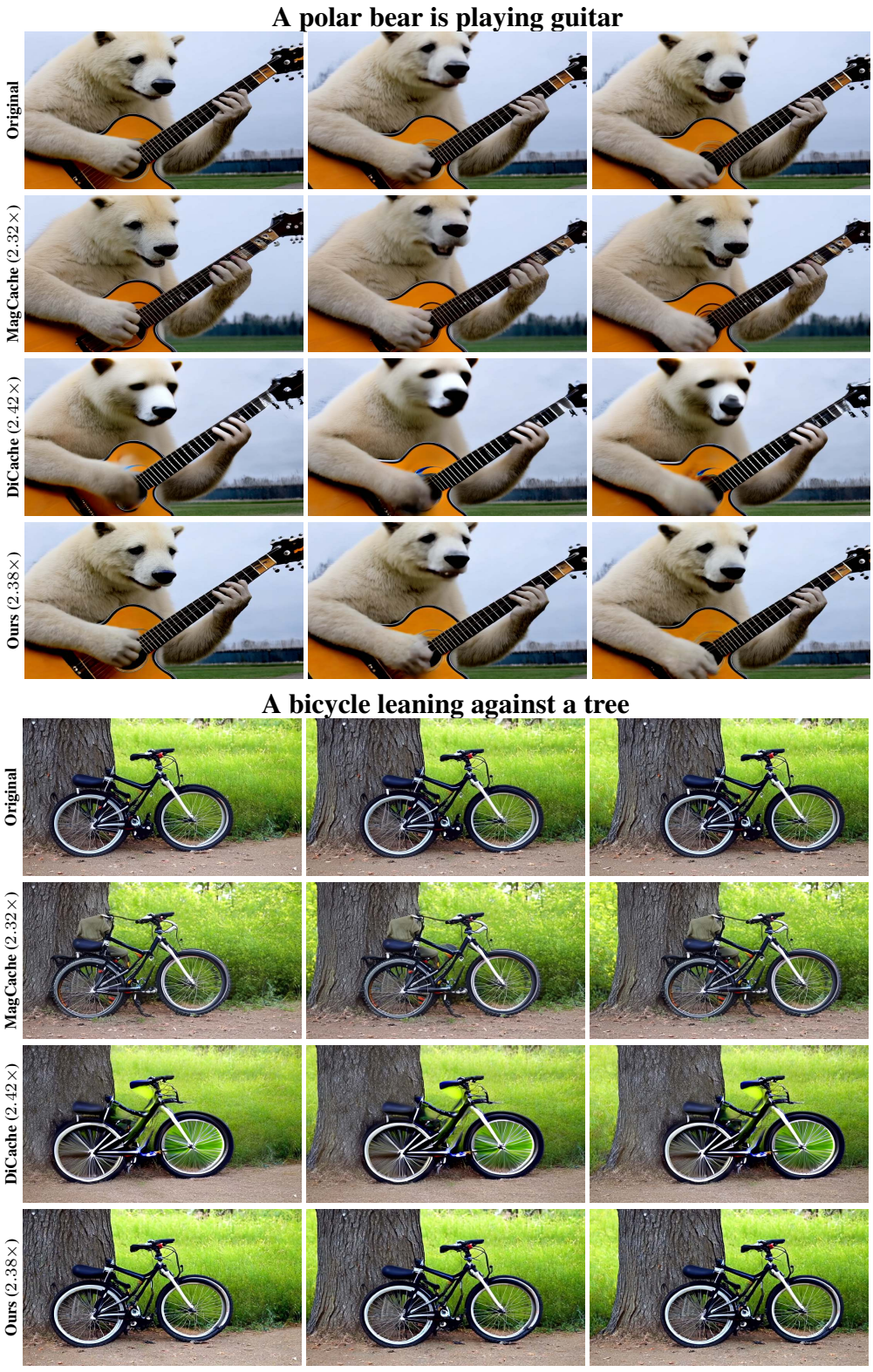


Figure 14: **Qualitative comparison on CogVideoX-2B (Part I).**



Figure 15: **Qualitative comparison on CogVideoX-2B (Part II).**

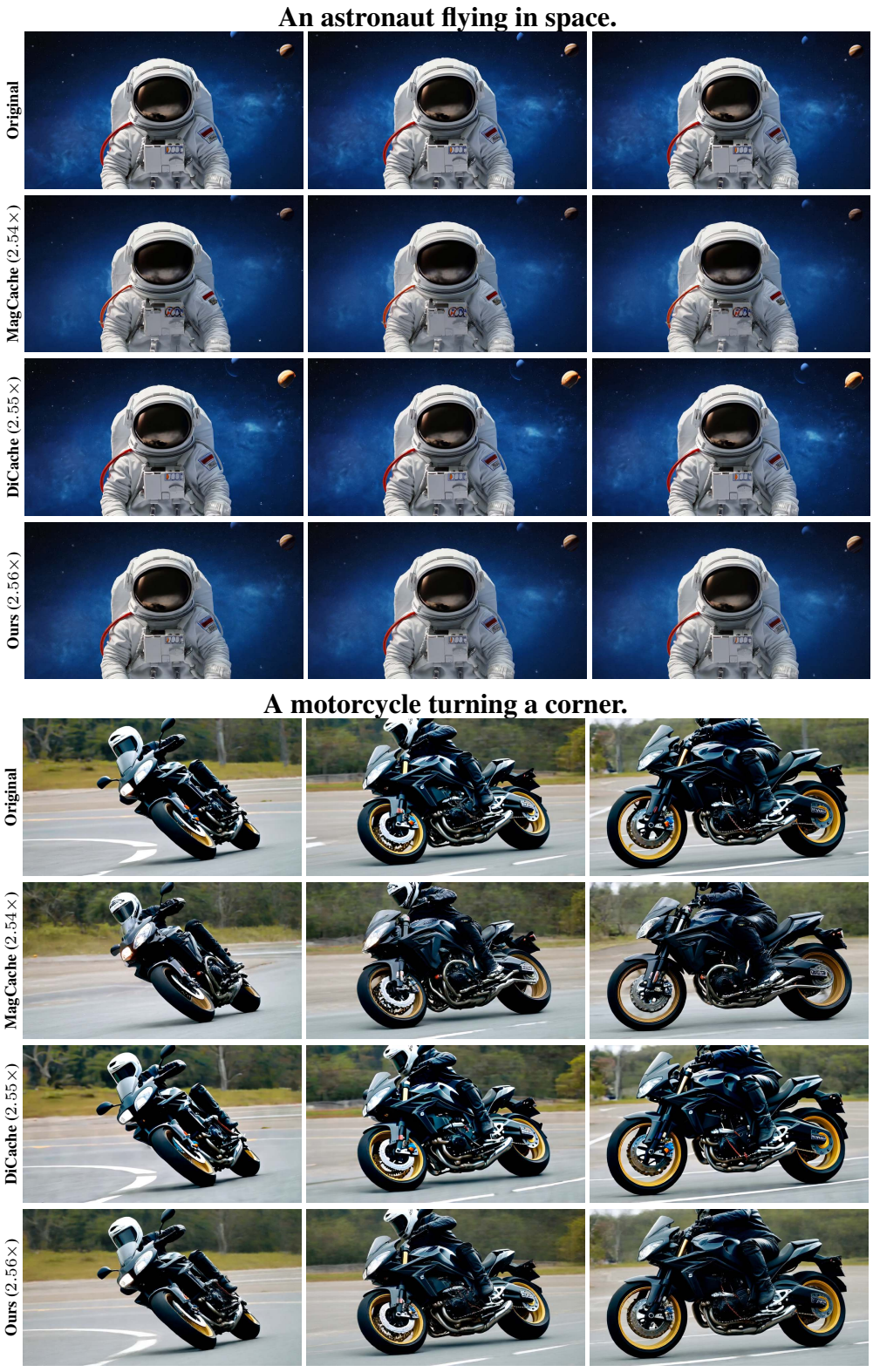


Figure 16: **Qualitative comparison on Wan2.1-T2V-1.3B (Part I).**

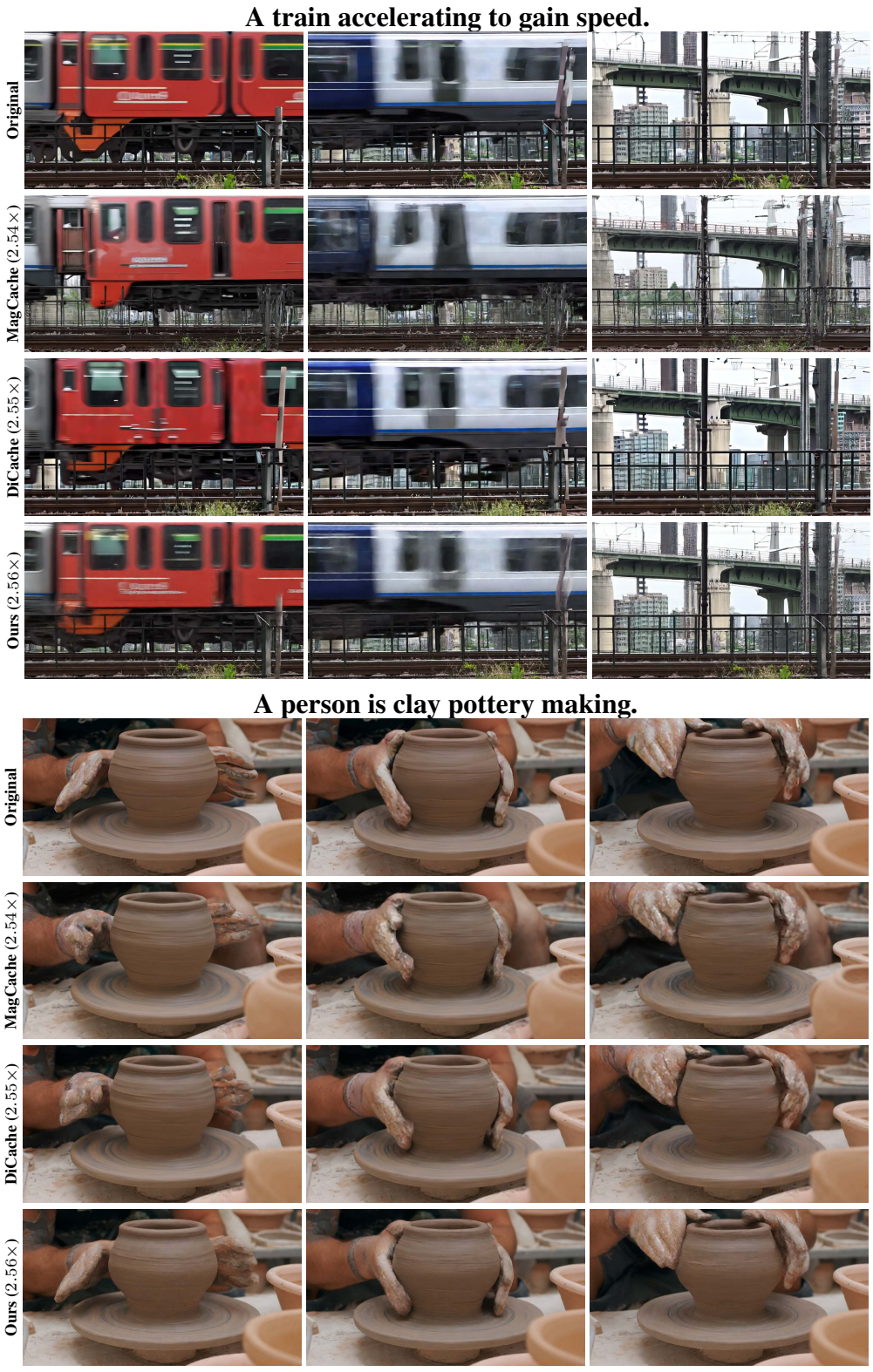


Figure 17: **Qualitative comparison on Wan2.1-T2V-1.3B (Part II).**

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 accurately state the scope and contributions of the paper, including the *branch–guided mismatch*, the *local–final mismatch*, and the proposed **RA-CFGCache** framework. These claims are supported by the analytical development in Section 3, the main and diagnostic experiments in Section 4, and the additional analyses in the appendices.

   Guidelines:

   - The answer NA 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 NA 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 paper discusses limitations in Appendix A, including the empirical and approximate nature of the propagation prior, the online approximation of the cumulative scheduler, and the reliance on configuration-specific offline calibration whose transfer across models, samplers, timestep schedules, or guidance settings may require validation or recalibration.

   Guidelines:

   - The answer NA 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 present formal theoretical theorems or proof-based guarantees. The analytical derivations in Sections 3 and Appendix C are used to motivate the proposed risk formulation and are supported by empirical calibration and ablation studies.

   Guidelines:

   - The answer NA 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 paper specifies the evaluated models, prompts, denoising steps, resolutions, CFG scales, baselines, metrics, and implementation settings in Section 4.1. Appendix B further provides the branch-wise proxy definitions, baseline configurations, and offline calibration procedure needed to reproduce **RA-CFGCache** and the main experimental comparisons.

   Guidelines:

   - The answer NA 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: The paper uses publicly available pretrained models and evaluation prompts, and we provide a public implementation and reproduction scripts through the repository linked in the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings are described in Section 4.1, including model checkpoints, inference steps, image/video resolutions, guidance scales, evaluation prompts, metrics, numerical precision, hardware, and CFG branch execution. Appendix B provides additional proxy and calibration details. Since **RA-CFGCache** is training-free, no optimizer or training split is required.

   Guidelines:

   - The answer NA 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 paper reports mean curves with $\pm$ one standard deviation across 100 prompts for the mechanism diagnostics in Fig. 3. However, the main end-to-end efficiency–fidelity comparisons are reported as aggregate metrics over fixed prompt sets without confidence intervals or formal statistical significance tests. We therefore conservatively answer No.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: Section 4.1 reports the GPU hardware, per-sample execution setup, and end-to-end latency measurements. Appendix B additionally reports the offline calibration complexity and compute cost; for example, the 100-prompt FLUX.1-dev calibration requires approximately 8.9 GPU-hours in total, together with the temporary storage requirements for calibration traces.

   Guidelines:

   - The answer NA 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 research conforms to the NeurIPS Code of Ethics. It studies a training-free inference acceleration algorithm for existing diffusion models, uses publicly available models and benchmark prompts for evaluation, and does not involve private data collection, human-subject experiments, crowdsourcing, or the release of new datasets or pretrained models.

Guidelines:

- The answer NA 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 paper discusses broader impacts in Appendix E. **RA-CFGCache** can lower deployment cost and reduce energy consumption, and we do not identify negative societal impacts unique to the caching algorithm; practical deployment should follow the safety policies of the underlying generative models.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new pretrained generative model, scraped dataset, or high-risk data asset. **RA-CFGCache** is an inference acceleration algorithm applied to existing models, whose deployment safeguards are governed by the original model providers and use policies.

Guidelines:

- The answer NA 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: [Yes]

    Justification: The paper cites the pretrained diffusion models, baseline acceleration methods, and evaluation benchmarks used in the experiments. Existing assets are used for research evaluation under their original licenses and terms of use.

    Guidelines:

    - The answer NA 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: We provide a public implementation and reproduction scripts for **RA-CFGCache**. The paper does not introduce a new dataset, benchmark, or pretrained model; the released code is documented for reproducing the main experiments.

    Guidelines:

    - The answer NA 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 paper does not involve crowdsourcing, user studies, or human-subject experiments. All evaluations are based on automatic metrics and qualitative comparisons.

Guidelines:

- The answer NA 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 paper does not involve human subjects or experiments requiring IRB approval.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as an important, original, or non-standard component of the proposed method or experiments. Any use of language tools, if any, is limited to writing, editing, or formatting and does not affect the scientific contribution.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.