SNAPCACHE: ACCELERATING FLOW MATCHING VIA TRAJECTORY PRUNING AND COMPENSATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow matching has emerged as a general method for generating images, videos, and actions. However, repeated sampling steps create a shared inference bottleneck. Step skipping reduces this cost by reusing cached velocities, but the resulting errors accumulate across repeated sampling steps. Model compression addresses a related problem by compensating for removed parameters through retained ones. We apply this principle to sampling with SNAPCACHE, a solver-agnostic step-skipping method that retains the original integration grid. At the next retained sampling step, a velocity residual reveals information about the error accumulated during the skipped interval without an additional model evaluation. Optimal Brain Trajectory Compensation (OBTC) estimates this error and applies a minimum-change output correction, paired with state injection for multistep sampling. We also develop a practical causal scheduler that selects the next skipped interval by comparing its predicted residual error with the solver's truncation-error scale. On FLUX.1-dev, Krea-2-Raw, and Wan2.2, SNAPCACHE improves sample-fidelity trade-offs over the compared caching baselines at approximately $2\times$–$4\times$ fewer network forward passes. On LIBERO-Long, action success reaches 94.0% at $6.32\times$ fewer model calls, compared with 94.5% for the full-compute policy.



Figure 1. Qualitative comparison at $2.00\times$–$5.56\times$ compression against a 50-NFE reference. LPIPS below each sample measures deviation (lower is better).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Flow matching generates images, videos, and actions (Esser et al., 2024; Wan Team, 2025; Black et al., 2024) by integrating learned velocity fields from noise to data (Lipman et al., 2023; Liu et al., 2023) through repeated sampling. Across different modalities, this process shares the problem of repeated velocity predictions, each of which requires a full model forward pass. Step caching and few-step distillation are well-known methods for reducing the cost of repeated sampling (Salimans & Ho, 2022; Song et al., 2023). Unlike distillation, step skipping reuses cached velocities or features (Ma et al., 2024; So et al., 2023; Selvaraju et al., 2024) without training, but the resulting errors propagate through the solver. Compensation has been actively studied in model compression. Optimal Brain Surgeon compensates for removed parameters by adjusting the retained ones (Hassibi & Stork, 1993), while Optimal Brain Compression extends this principle to quantization (Frantar et al., 2022). In contrast, cached reuse in sampling acceleration typically skips computation without explicitly compensating for the resulting errors, allowing them to accumulate along the sampling trajectory.

Can velocity/feature caching be viewed as a compensation problem?

This analogy views step skipping as *trajectory pruning*. SNAPCACHE retains the integration grid, reuses cached velocities at skipped timesteps, and compensates for their accumulated effect on the sampling trajectory (Figure 2).

Since the velocities at the start of a skipped interval and at the next retained step are already available, SNAPCACHE uses them as two anchors to estimate the accumulated sampling error without extra model foward pass. However, estimating this error alone does not correct the trajectory. The estimated error must be translated into a change in the solver state while disturbing the original solver dynamics as little as possible. To achieve this, we formulate trajectory correction as a minimum-change problem and introduce Optimal Brain Trajectory Compensation (OBTC), inspired by minimum-change pruning (Hassibi & Stork, 1993). For Euler, the accumulated skip error can be compensated by a single correction. For higher-order solvers such as UniPC, however, the correction must be distributed across multiple skipped steps. Given the estimated accumulated error, OBTC recovers the correction vectors for all skipped steps and uses them to update the solver buffer.

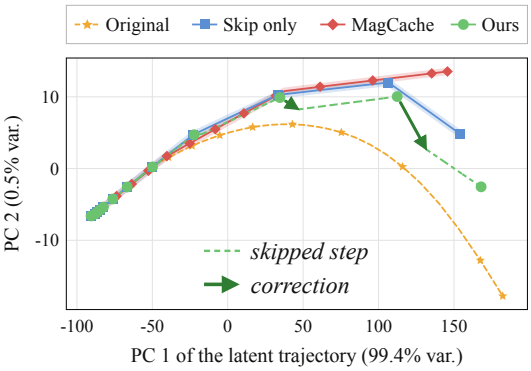

Figure 2. Projected sampling trajectories. Compensation reduces drift from the reference.

Existing step-skipping methods typically use fixed skip schedules that do not account for how sensitive the current sampling trajectory is to skipping. As a result, skip errors can accumulate and drive the trajectory away from the reference, as illustrated in Figure 2. Our *compensation-aware gate* adapts the skip length to the current trajectory using only past velocities. It compares the predicted residual error with the solver's truncation-error scale, allowing longer skips in smooth regions while remaining conservative when the trajectory is more sensitive.

Our contributions are as follows.

- We transfer prune-then-compensate from model compression to flow-matching sampling through a modality-agnostic, solver-preserving step-skipping framework.
- We derive a retrospective error estimator and solver-aware OBTC with a unique single-step correction and a multistep buffer update paired with state injection.
- We introduce a causal scheduler that budgets skip-induced error against solver truncation error using one dimensionless tolerance and no extra model evaluation.
- We demonstrate cross-modal applicability on FLUX.1-dev and Krea-2-Raw images, Wan2.2 videos, and LingBot-VA actions on LIBERO-Long, with solver-specific compensation under a common error-estimation and scheduling principle.

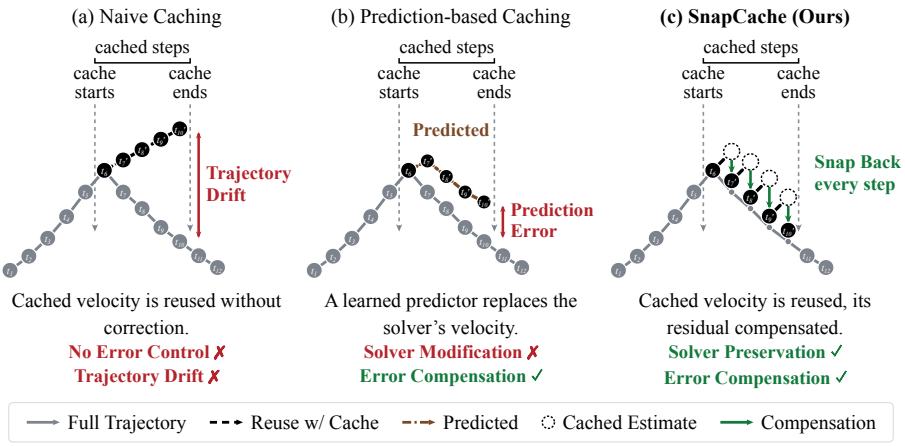

Figure 3. Caching strategies. Naive reuse accumulates drift; prediction changes the sampling trajectory; SNAPCACHE compensates retrospectively while preserving the solver.

2 RELATED WORK

Solver-based flow-matching inference. Standard flow-matching inference solves the ODE defined by the learned velocity field (Lipman et al., 2023; Liu et al., 2023). A numerical solver evaluates this field at discrete timesteps and integrates it from noise to data, approximating the continuous sampling trajectory. The solver therefore determines how velocity evaluations produce the generated sample. Related diffusion samplers also use numerical integration; DPM-Solver and UniPC improve its efficiency through specialized integration rules (Song et al., 2021; Lu et al., 2022; Zhao et al., 2023). Caching reduces per-step computation through temporal reuse. DeepCache and FRDiff reuse nearby features (Ma et al., 2024; So et al., 2023); FORA, Δ-DiT, TeaCache, FasterCache, and PAB extend reuse to transformer blocks and video (Selvaraju et al., 2024; Chen et al., 2024; Liu et al., 2024a; Lv et al., 2024; Zhao et al., 2024). Figure 3 compares how cached approximations affect the solver's trajectory and how the resulting error is handled.

Naive caching and uncontrolled drift (a). Reusing a cached velocity without correction introduces substitution errors that propagate through solver updates, producing the drift in panel (a). Reuse alone neither compensates for these errors nor explicitly controls them. Practical cache schedulers can restrict reuse to reduce exposure, but this does not repair accumulated error.

Prediction-based caching (b). TaylorSeer forecasts cached features through Taylor expansions (Liu et al., 2025). FoCa models feature evolution as an ODE and uses forecast-then-calibrate updates (Zheng et al., 2025), while Confidence-Gated Taylor uses first-block prediction error to decide whether to forecast or perform full computation (Guan et al., 2025). Panel (b) schematically depicts the resulting approximation to the velocity supplied to the sampler, which need not change the external solver's grid or update rule. Improving feature forecasts does not by itself explicitly correct the accumulated sampling-state discrepancy. Our focus is this propagated error, rather than prediction of missing features alone.

Compensation with solver preservation (c). Optimal Brain Surgeon and Optimal Brain Compression compensate for pruning or quantization through retained parameters (Hassibi & Stork, 1993; Frantar et al., 2022). SNAPCACHE transfers this principle to skipped sampling steps. As shown in panel (c), the original solver grid and update rule are retained, while a retained sampling step reveals a residual used to estimate and compensate the completed skip interval's accumulated sampling error. Compensation acts on retained outputs or history entries, and a causal gate controls future step skipping. Solver preservation therefore means retaining the numerical integration scheme, not reproducing the full-step sampling trajectory exactly; the remaining approximation error is analyzed in Section 3.

3 METHOD

3.1 STEP SKIPPING AND ACCUMULATED ERROR

We formulate step skipping directly at the level of numerical integration. Skipping a step removes the model forward pass while preserving the solver update.

$$x_{k+1} = \Phi_k(x_k; u_k, u_{k-1}, \ldots), \qquad u_k = u_\theta(x_k, t_k). \tag{1}$$

Here x_k is the sampling state at grid time t_k, u_θ is the learned velocity field, and Φ_k is the solver update. Consider two consecutive retained indices $s < r$. We call the interval between them a *skip interval*. These skipped steps reuse $\tilde{u} = u_\theta(x_s, t_s)$ until the next retained index r. The hidden error at a skipped index is $\delta_i = u_\theta(x_i^{\text{skip}}, t_i) - \tilde{u}$; computing it exactly would forfeit the saving.

For local continuations with the same initial state and solver history, we separate the direct solver contribution from state feedback.

$$x_r^{\text{full}} - x_r^{\text{skip}} = \underbrace{\sum_{i \in \mathcal{S}} A_i \delta_i}_{e} + R_{\text{state}}, \tag{2}$$

Here $\mathcal{S}$ is the set of skipped indices. The solver sensitivity $A_i = \partial x_r / \partial u_i$ holds all other supplied outputs fixed; it is not the total sensitivity of the neural sampling trajectory. For linear-in-output solvers, it depends only on solver coefficients and the grid. We estimate the direct contribution $e \equiv e_{\text{solver}}$, not R_{state}, which captures subsequent output changes induced by state feedback (Appendix A).

3.2 RETROSPECTIVE ERROR ESTIMATION

The scheduled refresh reveals the velocity's change from its cached value, providing two reconstruction anchors without an extra network call.

$$\nu = u_r - \tilde{u}, \qquad \delta(t_s) = 0, \quad \delta(t_r) = \nu. \tag{3}$$

Here $u_r = u_\theta(x_r^{\text{skip}}, t_r)$ is the refreshed velocity. The residual ν provides retrospective information about the completed skip interval, rather than predicting a future velocity.

The simplest reconstruction connects these anchors with a linear ramp. Weighting this ramp by the solver sensitivities in Equation 2 gives

$$\hat{e} = B_{\mathcal{S}} \nu, \qquad B_{\mathcal{S}} = \sum_{i \in \mathcal{S}} A_i \rho(t_i). \tag{4}$$

Here $\rho(t) = (t - t_s)/(t_r - t_s)$ defines the ramp $\hat{\delta}(t) = \rho(t)\nu$, and $B_{\mathcal{S}}$ combines its contributions to the endpoint. This matrix is precomputable for a fixed solver and grid, so one observed vector estimates the aggregate error.

The linear ramp captures first-order variation but misses curvature. For a twice-differentiable error trajectory, the interpolation remainder therefore bounds the reconstruction error of the direct solver contribution.

$$\|e - \hat{e}\| \le \underbrace{\frac{1}{2} \sum_{i \in \mathcal{S}} \|A_i\| \, |(t_i - t_s)(t_i - t_r)|}_{C_{\mathcal{S}}} \sup_{t \in [t_s, t_r]} \left\| \ddot{\delta}(t) \right\|. \tag{5}$$

Here $\ddot{\delta}(t)$ measures error-trajectory curvature and $C_{\mathcal{S}}$ collects the solver and time-grid weights. Longer skipped intervals amplify the remaining error when a linear ramp cannot follow this curvature.

3.3 SOLVER-AGONISTIC OPTIMAL BRAIN TRAJECTORY COMPENSATION

Multiple combinations of corrections to the retained velocity predictions can induce the same estimated endpoint correction. We resolve this ambiguity by selecting the minimum-norm set of velocity corrections whose first-order endpoint effect matches $\hat{e}$..

$$c_{\text{OBTC}} = \underset{c,\ Ac = \hat{e}}{\arg\min} \ \tfrac{1}{2} \|c\|_2^2 = A^\top (A A^\top)^{-1} \hat{e}, \tag{6}$$

Here $c = [c_1^\top \ \cdots \ c_M^\top]^\top$ stacks corrections to M retained variables, and $A = [A_1 \ \cdots \ A_M]$ stacks their endpoint sensitivities. Thus $Ac = \hat{e}$ enforces compensation; the closed form assumes full row rank. This identity-metric OBTC update is orthogonal to $\ker(A)$, adding no component unsupported by the observation.

Euler multiplies each velocity by its step size. Compensating for the estimated state error therefore requires dividing it by that step size and adding it to the refreshed velocity.

$$u_r^{\text{ret}} = u_\theta(x_r^{\text{skip}}, t_r) + \hat{e}/h_r. \tag{7}$$

Here $h_r = t_{r+1} - t_r$ is the Euler step size and u_r^{ret} is the corrected velocity. Since the refreshed velocity enters the Euler update with sensitivity $h_r I$, this single-variable correction is unique. For UniPC, the predictor reads both the current sample and stored model outputs. Our final recipe combines state injection with buffer-OBS, the history correction from Equation 6. They repair complementary inputs, rather than replacing one another. Correcting only one leaves the predictor mixing corrected and stale quantities. Appendix B gives the history sensitivities; Section 4.7 tests the combined recipe.

3.4 Causal adaptive scheduling

Scheduling precedes skipping, so the future refresh residual cannot be an input. We estimate local derivatives from velocity differences available at the current retained step and freeze them over the candidate interval; grid-dependent factors are precomputed.

Skipping should stop before the error remaining after compensation exceeds the solver's own error scale. We therefore choose the longest candidate interval whose predicted residual fits a time-dependent truncation-error budget.

$$m^* = \max\{m \mid E_{\text{cache}}(m) \le \kappa\, g(\tau)\, E_{\text{solver}}(m)\}, \qquad g(\tau) = 4\tau(1-\tau), \tag{8}$$

Here $m = r - s$ is the candidate skip length, subject to a fixed maximum, $E_{\text{cache}}(m)$ is the predicted residual-error scale derived from Equation 5, and $E_{\text{solver}}(m)$ is the solver's leading truncation-error scale over the same interval. Here $\tau \in [0, 1]$ is normalized sampling time and κ is dimensionless because both sides compare state errors. The endpoint weight is conservative where early errors propagate far or late errors have few chances for compensation.

The next refresh compensates for the completed interval and updates the derivative estimates. Appendices A and B give the Euler and UniPC error scales.

3.5 Complete procedure

Algorithm 1 alternates retrospective compensation and causal scheduling after initializing the velocity history.

Algorithm 1 Adaptive step skipping with SnapCache

Require initial state x_0, grid $\{t_k\}$, solver Φ, tolerance κ

1: Compute enough initial outputs to initialize the solver and derivative estimates.
2: Evaluate the velocity at step s and set $\tilde{u} = u_\theta(x_s, t_s)$.
3: **while** s is not the final solver index **do**
4: Estimate available output derivatives from computed history.
5: Choose m^* with the causal gate in Equation 8; set $r = s + m^*$.
6: Skip velocity evaluations at $s+1, \ldots, r-1$ and advance them with $\tilde{u}$.
7: Evaluate the velocity at step r; compute $u_r = u_\theta(x_r^{\text{skip}}, t_r)$ and $\nu = u_r - \tilde{u}$.
8: Estimate the accumulated error $\hat{e} = B_{\mathcal{S}}\nu$.
9: Apply output compensation for Euler; apply state injection and buffer compensation for UniPC.
10: Set $s \leftarrow r$ and $\tilde{u} \leftarrow u_r + c_r$, where c_r corrects the refreshed output.
11: **end while**
12: **return** final state

The scheduler and OBTC use derivative estimates and small output or history updates, with no training, calibration set, or additional model evaluation.

Table 1. Image fidelity on GenEval-553. Sky-blue rows denote SNAPCACHE; bold marks best fidelity within each band, excluding Full.

Method	Calibration-free	Kept ↓	Comp ↑	LPIPS ↓	SSIM ↑	PSNR ↑	Ref-FID ↓
FLUX.1-dev, 1024 × 1024, *50 steps*							
Full sampler	–	50.00	1.00	–	–	–	–
TeaCache	✗	27.04	1.85	0.1887	0.8201	21.43	23.92
ERTACache	✗	25.00	2.00	0.1546	0.8468	22.96	20.52
MagCache	✗	24.00	2.08	**0.0561**	0.9329	29.55	**8.17**
SNAPCACHE	✓	21.72	2.30	0.0562	**0.9336**	**29.57**	8.68
TeaCache	✗	12.97	3.85	0.3952	0.6736	16.64	43.55
ERTACache	✗	12.00	4.17	0.4515	0.6990	19.62	49.10
MagCache	✗	11.00	4.55	0.5686	0.6300	21.62	67.15
SNAPCACHE	✓	11.29	4.43	**0.1995**	**0.8087**	**22.71**	**24.53**
Krea-2-Raw, 1024 × 1024							
Full sampler	–	52.00	1.00	–	–	–	–
ERTACache	✗	21.00	2.48	0.2975	0.7380	18.39	45.79
MagCache	✗	21.00	2.48	0.1355	0.8523	23.85	27.20
TeaCache	✗	19.00	2.74	0.3340	0.7137	17.20	47.54
SNAPCACHE	✓	20.54	2.53	**0.116**	**0.880**	**25.18**	**23.33**
ERTACache	✗	14.00	3.71	0.5503	0.5970	16.86	108.84
MagCache	✗	13.00	4.00	0.4421	0.6591	19.64	93.86
TeaCache	✗	12.00	4.33	0.4499	0.6419	15.08	62.49
SNAPCACHE	✓	12.91	4.03	**0.2519**	**0.7634**	**19.90**	**38.93**

4 EXPERIMENTS

4.1 SAMPLING SETUP

Experiments test the shared flow-matching framework across image fidelity, video fidelity, and action task success, rather than assuming a common quality metric. We compare MagCache (Ma et al., 2025), ERTACache (Peng et al., 2025), and TeaCache (Liu et al., 2024a). Kept counts fresh velocity predictions; compression is nominal steps divided by Kept, not latency. Krea uses two transformer calls per kept step under true CFG. Only identical Kept counts constitute equal-budget comparisons. Bold marks the best quality score among accelerated methods within each table band, excluding full-compute references. Calibration-free means no pre-calibration; the full sampler is not applicable.

FLUX.1-dev (Black Forest Labs, 2024) uses 50 Euler steps, seed 0, and 553 GenEval prompts (Ghosh et al., 2023); Krea-2-Raw (Krea AI, 2026b) uses 52 steps. Both use 1024^2 images and guidance 3.5. Table 1 selects each method's nearest measured setting to $2\times$ and $4\times$ for FLUX, and $2.5\times$ and $4\times$ for Krea, independently of quality scores. LPIPS, SSIM, PSNR, and Ref-FID compare accelerated samples with prompt- and seed-matched full-step references.

4.2 IMAGE SAMPLING FIDELITY

On FLUX (Table 1), SNAPCACHE at 21.72 kept steps nearly matches MagCache's LPIPS at 24 (0.0562 versus 0.0561), with higher SSIM and PSNR. At 11.29 steps, it improves all reported fidelity metrics over ERTACache at 12 and TeaCache at 12.97. On Krea, SNAPCACHE at 20.54 improves fidelity over the listed baselines. At 12.91, it improves all fidelity metrics over MagCache at 13. The selected rows compare nearby operating points, not uniformly matched budgets.

4.3 VIDEO SAMPLING FIDELITY

Table 2. Wan2.2 fidelity at 1280×704. Sky-blue rows denote SNAPCACHE.

Method	Calibration-free	Kept ↓	Comp ↑	LPIPS ↓	SSIM ↑	PSNR ↑	VBench Quality ↑ (%)
Full sampler	–	50.00	1.00	–	–	–	82.82
MagCache	✗	24.00	2.08	0.0370	0.9448	30.47	82.46
TeaCache	✗	24.00	2.08	0.0415	0.9381	30.05	82.38
SNAPCACHE	✓	21.20	2.36	**0.0276**	**0.9543**	**32.07**	**82.52**
ERTACache	✗	18.00	2.78	0.2091	0.7788	20.59	**82.31**
TeaCache	✗	18.00	2.78	0.4434	0.6087	15.29	81.92
SNAPCACHE	✓	18.21	2.75	**0.0472**	**0.9209**	**29.16**	82.29
ERTACache	✗	12.00	4.17	0.2577	0.7405	20.16	81.88
MagCache	✗	12.00	4.17	0.2269	0.7522	22.17	81.96
TeaCache	✗	11.00	4.55	0.5398	0.5807	14.50	81.31
SNAPCACHE	✓	11.76	4.25	**0.1401**	**0.8147**	**23.28**	**82.05**

Protocol and reporting scope. Wan2.2-TI2V-5B (Wan Team, 2025) uses a 50-step UniPC reference, 1280×704 resolution, 100 VBench prompts, seed 42, and flow shift 5. The final SNAPCACHE recipe combines buffer compensation and state injection.

Nearby-budget comparisons. At 18.21 kept steps, SNAPCACHE reaches LPIPS 0.0472 versus 0.2091 for ERTACache and 0.4434 for TeaCache at 18. At 11.76 steps, it reaches 0.1401, versus 0.2269 for MagCache and 0.2577 for ERTACache at 12, and 0.5398 for TeaCache at 11. These comparisons use similar, not identical, NFE. Figure 4 shows paired samples from the qualitative study. VBench Quality is the normalized five-dimension aggregate, shown as a percentage. Across the three bands, SNAPCACHE scores 82.52%, 82.29%, and 82.05%, compared with 82.82% for the full sampler.

4.4 INTERPRETING THE QUALITY–COMPUTE TRADE-OFF

At nearby budgets, SNAPCACHE reduces Wan LPIPS by 77.4% relative to 18-step ERTACache, using 1.2% more NFE. Other pairs improve both quality and computation. Against TeaCache on FLUX, it uses 13.0% fewer forwards with 49.5% lower LPIPS; against 12-step MagCache on Wan, it uses 2.0% fewer forwards with 38.3% lower LPIPS. These are comparisons of the listed operating points, not an interpolated frontier. Table 4 isolates compensation with a fixed gate and tolerance.

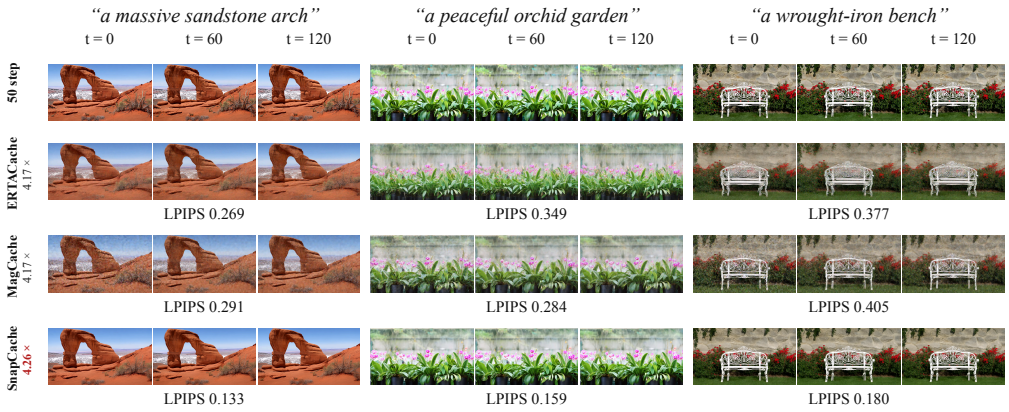

Figure 4. Wan2.2 video comparisons at approximately $4\times$ compression. Frames at $t = 0, 60, 120$ and LPIPS show fidelity to the 50-step reference.

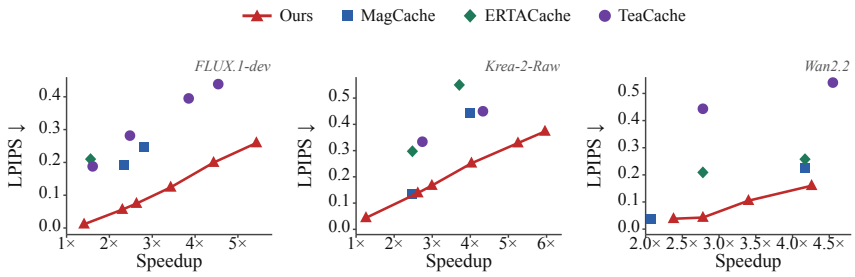

Figure 5. Sample fidelity versus reported speedup across three models. Varying κ adjusts SNAPCACHE's operating point without additional pre-calibration.

4.5 ACTION SAMPLING FOR ROBOT CONTROL

LingBot-VA task success. Table 3 uses the public LIBERO-Long checkpoint (Li et al., 2026). The lower-compression study has 10 tasks with 50 episodes each. Caching applies to both video and action branches; Full uses 72 transformer forwards per chunk, comprising 21 video and 51 action calls. Compression is $72/\text{Calls}$, making this first band approximately $2.2\times$. TeaCache, Block-wise Adaptive Caching (BAC) (Ji et al., 2025), and SNAPCACHE achieve 94.4–94.8% success.

Table 3. LingBot-VA on LIBERO-Long. Calls count transformer forwards per chunk.

Method	Calls ↓	Comp ↑	Success (%)	n
Approximately $2.2\times$*, 500 episodes*				
Full	72.00	1.00	94.4	500
TeaCache	31.88	2.26	94.4	500
BAC	34.16	2.11	94.6	500
SNAPCACHE	32.93	2.19	**94.8**	500
Approximately $6.3\times$*, 200 episodes*				
Full	72.00	1.00	94.5	200
TeaCache	10.95	6.58	87.0	200
SNAPCACHE	11.39	6.32	**94.0**	200

At approximately $6.3\times$, the 200-episode study separates the methods. SNAPCACHE achieves 94.0% versus 87.0% for TeaCache and 94.5% for Full. Reported paired exact McNemar tests give $p = 0.024$ for SNAPCACHE versus TeaCache, $p = 0.0059$ for TeaCache versus Full, and $p = 1.00$ for SNAPCACHE versus Full. Thus no difference from Full is detected for SNAPCACHE at this sample size; this is not an equivalence claim.

LIBERO qualitative example. Figure 6 shows one illustrative rollout per method. The figure labels task completion at 159 s for the reference and 72 s for SNAPCACHE, and failure at 192 s for TeaCache. The action factors marked $4.21\times$ and $4.11\times$ are distinct from elapsed-time ratios. This example illustrates a task outcome, not a benchmark-wide success rate or a general latency improvement.

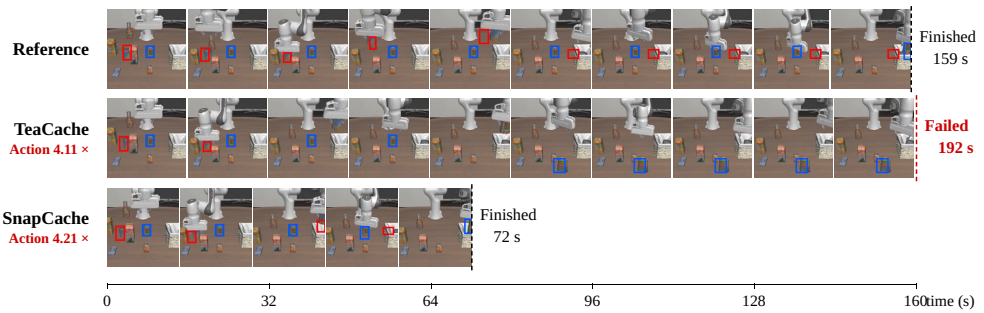

Figure 6. LIBERO rollout comparison. The reference and SNAPCACHE complete the task, while TeaCache fails; the horizontal axis shows elapsed time.

4.6 SAMPLING QUALITY ACROSS COMPRESSION LEVELS

Figure 5 shows quality–efficiency trade-offs controlled by κ, with online adaptation and no additional pre-calibration or retraining. Tables 2 and 4 report the final combined Wan recipe.

From NFE reduction to sampling latency. For compression C, baseline forward-pass time fraction f, and normalized overhead η, unchanged per-forward-pass cost gives the idealized speedup $S_{\text{time}} = [(1-f) + f/C + \eta]^{-1}$. Compression alone therefore does not determine runtime. Robot completion times also include environment interaction, rather than sampler latency alone.

4.7 EFFECT OF SOLVER-AWARE COMPENSATION

Table 4. Wan2.2 compensation ablation with a shared gate and tolerance. Parentheses show relative changes from Off (%); PSNR changes use its dB values.

Band	Compensation	Kept ↓	Comp ↑	LPIPS ↓	SSIM ↑	PSNR ↑
2.75×	Off (gate only)	18.09	2.76	0.0652	0.9143	28.68
2.75×	On (ours)	18.21	2.75	**0.0472** (−27.6%)	**0.9209** (+0.72%)	**29.16** (+1.67%)
4.25×	Off (gate only)	11.77	4.25	0.1649	0.8109	23.20
4.25×	On (ours)	11.76	4.25	**0.1401** (−15.0%)	**0.8147** (+0.47%)	**23.28** (+0.34%)

Table 4 uses Wan2.2-TI2V-5B with a 50-step UniPC reference at 1280×704, 100 VBench prompts, seed 42, and flow shift 5. With the same LTE gate and tolerance, compensation reduces LPIPS by 27.6% and 15.0%. Kept differs by 0.12 and 0.01, so budgets are nearly matched, not identical.

State injection and buffer-OBS repair complementary UniPC inputs, the current sample and output history; Appendix E details their individual effects.

4.8 ACCELERATING FEW-STEP DISTILLED MODELS

Distillation (Liu et al., 2024b; Luo et al., 2025) and step skipping reduce sampling cost in complementary ways. On Krea-2-Turbo (Krea AI, 2026b;a) (Table 5), SNAPCACHE preserves fidelity better than the caching baselines across all three metrics at both sampling budgets. The advantage persists under more aggressive skipping, despite the already short distilled trajectory. These results suggest that solver-aware compensation remains useful after distillation, although skipping more steps still reduces fidelity.

Table 5. Krea-2-Turbo acceleration. Fidelity is measured against its 8-step reference.

Method	Kept ↓	LPIPS ↓	SSIM ↑	PSNR ↑
5 retained steps (≈ 1.60×)				
TeaCache	4.96	0.381	0.787	23.93
MagCache	5.00	0.247	0.818	20.57
SNAPCACHE	4.99	**0.098**	**0.905**	**25.45**
3 retained steps (2.67×)				
TeaCache	3.00	0.597	0.528	13.92
MagCache	3.00	0.526	0.607	15.41
SNAPCACHE	3.00	**0.340**	**0.733**	**18.95**

5 CONCLUSION

SNAPCACHE accelerates flow-matching sampling across images, videos, and actions through residual-based error estimation, solver-aware compensation, and causal step skipping. Experiments support this shared mechanism on the tested models and show that it can also complement few-step distillation. Rather than treating skipped predictions as independent omissions, the method targets their accumulated effect through the numerical solver. Euler and UniPC require different corrections, but share the same error-estimation and adaptive scheduling principle. Image and video results demonstrate improved fidelity at nearby sampling budgets, while LIBERO-Long retains 94.0% task success at 6.32× fewer model calls. A single tolerance adjusts the quality–compute trade-off without additional pre-calibration. Together, these results support solver-aware compensation as a shared route to efficient flow-matching sampling across modalities.

Reproducibility Statement

The mathematical assumptions, accumulated-error estimator, OBTC rule, and causal scheduler are specified in Section 3; solver-specific derivations appear in Appendices A and B. Appendix E lists the sampling configurations, metric semantics, and data exclusions used for the reported results. The source code is included in the supplementary material.

AI Use Statement

We used generative AI tools to assist with grammar, language editing, and sentence-level phrasing. We did not use generative AI tools for drafting or finalizing the manuscript. All research ideas and experimental designs were developed without the use of generative AI. Generative AI tools were also used to assist with verifying mathematical derivations related to the experiments. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final content of this work.

References

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Black Forest Labs. FLUX.1: Official inference repository. https://github.com/black-forest-labs/flux, 2024.

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. *arXiv preprint arXiv:2406.01125*, 2024. URL https://arxiv.org/abs/2406.01125.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2403.03206.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *arXiv preprint arXiv:2208.11580*, 2022.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2310.11513.

Xiaoliu Guan, Lielin Jiang, Hanqi Chen, Xu Zhang, Jiaxing Yan, Guanzhong Wang, Yi Liu, Zetao Zhang, and Yu Wu. Forecasting when to forecast: Accelerating diffusion models with confidence-gated taylor. *arXiv preprint arXiv:2508.02240*, 2025. URL https://arxiv.org/abs/2508.02240.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pp. 164–171, 1993. URL https://proceedings.neurips.cc/paper_files/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf.

Kangye Ji, Yuan Meng, Hanyun Cui, Ye Li, Jianbo Zhou, Shengjia Hua, Lei Chen, and Zhi Wang. Block-wise adaptive caching for accelerating diffusion policy. *arXiv preprint arXiv:2506.13456*, 2025. URL https://arxiv.org/abs/2506.13456.

Krea AI. Krea 2: Official inference code and checkpoints. https://github.com/krea-ai/krea-2, 2026a.

Krea AI. Krea 2 technical report. https://www.krea.ai/blog/krea-2-technical-report, 2026b.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. URL https://arxiv.org/abs/2601.21998.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

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. *arXiv preprint arXiv:2411.19108*, 2024a. URL https://arxiv.org/abs/2411.19108.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. *arXiv preprint arXiv:2503.06923*, 2025. URL https://arxiv.org/abs/2503.06923.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.03003.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation. In *International Conference on Learning Representations*, 2024b. URL https://arxiv.org/abs/2309.06380.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2206.00927.

Yihong Luo, Tianyang Hu, Jiacheng Sun, Yujun Cai, and Jing Tang. Learning few-step diffusion models by trajectory distribution matching. In *IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2503.06674.

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. *arXiv preprint arXiv:2410.19355*, 2024. URL https://arxiv.org/abs/2410.19355.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2312.00858.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. MagCache: Fast video generation with magnitude-aware cache. *arXiv preprint arXiv:2506.09045*, 2025. URL https://arxiv.org/abs/2506.09045.

Xurui Peng, Hong Liu, Chenqian Yan, Rui Ma, Fangmin Chen, Xing Wang, Zhihua Wu, Songwei Liu, and Mingbao Lin. ERTACache: Error rectification and timesteps adjustment for efficient diffusion. *arXiv preprint arXiv:2508.21091*, 2025. URL https://arxiv.org/abs/2508.21091.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2202.00512.

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

Junhyuk So, Jungwon Lee, and Eunhyeok Park. FRDiff: Feature reuse for universal training-free acceleration of diffusion models. *arXiv preprint arXiv:2312.03517*, 2023. URL https://arxiv.org/abs/2312.03517.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Wan Team. Wan2.2: Open and advanced large-scale video generative models. https://github.com/Wan-Video/Wan2.2, 2025.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2302.04867.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024. URL https://arxiv.org/abs/2408.12588.

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. *arXiv preprint arXiv:2508.16211*, 2025. URL https://arxiv.org/abs/2508.16211.

A Euler Instantiation

Consider the Euler update $x_{k+1} = x_k + h_k u_k$. We treat the supplied model outputs as independent inputs to the fixed-grid solver. Holding all other supplied outputs fixed, a perturbation δ_i to output u_i changes the endpoint by $h_i\delta_i$. Thus the direct solver sensitivity is $A_i^{\text{solver}} = h_i I$.

Let $\delta_i = u_\theta(x_i^{\text{skip}}, t_i) - \tilde{u}$. We approximate the accumulated trajectory discrepancy by the direct solver contribution

$$e_{\text{solver}} = \sum_{i \in \mathcal{S}} h_i \delta_i.$$

This approximation neglects changes in subsequent network outputs induced by changes in the sampling state. Using the zero-anchor linear residual reconstruction of Section 3.2 gives

$$\hat{e} = \frac{W_1}{\Delta}\nu, \qquad W_1 = \sum_{i \in \mathcal{S}} h_i(t_i - t_s), \quad \Delta = t_r - t_s. \tag{9}$$

On a uniform grid, this is the discrete analogue of the triangular area under the reconstructed residual.

For a local comparison, let the full and skipped continuations start from the same state immediately after the retained update at index s. Their endpoint discrepancy satisfies

$$x_r^{\text{full}} - x_r^{\text{skip}} = e_{\text{solver}} + R_{\text{state}},$$

where

$$R_{\text{state}} = \sum_{i \in \mathcal{S}} h_i \left[u_\theta(x_i^{\text{full}}, t_i) - u_\theta(x_i^{\text{skip}}, t_i) \right].$$

Our estimator targets e_{solver} without evaluating the state-feedback term R_{state}.

Keeping the refreshed network evaluation fixed, adding $c_r = \hat{e}/h_r$ to its output changes the next Euler state by exactly $\hat{e}$, yielding Equation 7. This compensates the estimated direct solver discrepancy; it does not imply exact recovery of the full-compute trajectory.

For scheduling, a Taylor expansion of one Euler step gives a leading local solver error proportional to $h_k^2 \|\dot{u}\|$. Define, for candidate skip interval $B(m)$ and skipped indices $\mathcal{S}(m)$,

$$W_1(m) = \sum_{i \in \mathcal{S}(m)} h_i(t_i - t_s), \qquad W_2(m) = \sum_{i \in \mathcal{S}(m)} h_i(t_i - t_s)^2, \tag{10}$$

$$\Delta(m) = t_{s+m} - t_s. \tag{11}$$

The two scales used by the gate are

$$E^{(1)}_{\text{cache}}(m) = \tfrac{1}{2}\,\|\ddot{u}\|\;|W_1(m)\Delta(m) - W_2(m)|, \tag{12}$$

$$E^{(1)}_{\text{solver}}(m) = \tfrac{1}{2}\,\|\dot{u}\| \sum_{j\in B(m)} h_j^2. \tag{13}$$

The derivative norms are estimated at the current retained step from computed outputs. Inserting these terms into Equation 8 gives an entirely causal Euler scheduler. On a uniform grid the skip-error residual grows faster with skip length than the solver budget, which guarantees a finite largest admissible candidate and yields square-root growth in κ under the local smoothness approximation. The gate compares causal estimates of the residual-reconstruction error and the solver's local truncation-error scale. It does not bound the unmodeled state-feedback term.

B UniPC Instantiation

After expressing a selected UniPC configuration in the network-output variable u, its update can be written

$$x_{k+1} = P_k x_k + \sum_{\ell=0}^{p-1} \beta_{k,\ell} u_{k-\ell}, \tag{14}$$

where P_k and $\beta_{k,\ell}$ are fixed by the order, parameterization, and grid. The linearity in buffered outputs is the property needed to precompute sensitivities. Let $G_{k,a} = \partial x_k/\partial u_a$. Then

$$G_{k+1,a} = P_k G_{k,a} + \mathbf{1}\{0 \le k-a < p\}\,\beta_{k,k-a} I, \qquad G_{a,a} = 0. \tag{15}$$

For a candidate endpoint $s+m$, SnapCache uses $A^{\text{UniPC}}_a(m) = G_{s+m,a}$. At the next retained sampling step, the sensitivities of affected history entries form A in Equation 6. This describes the buffer component, not the entire combined recipe. In the final Wan adapter, state injection also corrects the sample passed to the predictor. Since Equation 14 depends on both x_k and buffered outputs, correcting just one input leaves the other stale. The buffer-only minimum-norm derivation does not by itself characterize the joint update.

Applying Equation 5 gives the predicted skip-error scale

$$E^{\text{UniPC}}_{\text{cache}}(m) = \tfrac{1}{2}\widehat{\|\ddot{u}\|}_s \sum_{a\in\mathcal{S}(m)} \left\|A^{\text{UniPC}}_a(m)\right\|\,|(t_a - t_s)(t_a - t_{s+m})|. \tag{16}$$

To obtain the solver scale, write the exact one-step evolution in mild form as $x(t_{k+1}) = P_k x(t_k) + \int_{t_k}^{t_{k+1}} K_k(t)u(t)\,dt$. Define the exact and numerical derivative moments

$$\mu_{k,q} = \frac{1}{q!}\int_{t_k}^{t_{k+1}} K_k(t)(t-t_k)^q\,dt, \qquad \tilde{\mu}_{k,q} = \frac{1}{q!}\sum_{\ell=0}^{p-1}\beta_{k,\ell}(t_{k-\ell}-t_k)^q. \tag{17}$$

If q is the convergence order, the first unmatched moment has magnitude $D_{k,q} = \|\mu_{k,q} - \tilde{\mu}_{k,q}\|$. With $T_{s+m\leftarrow k+1} = P_{s+m-1}\cdots P_{k+1}$, the propagated candidate-interval scale is

$$E^{\text{UniPC}}_{\text{solver}}(m) = \widehat{\left\|u^{(q)}\right\|}_s \sum_{k\in B(m)} \|T_{s+m\leftarrow k+1}\|\,D_{k,q}. \tag{18}$$

All coefficients, propagation factors, and sensitivities are precomputed. The derivative estimates come from UniPC outputs available at t_s and remain frozen for the candidate skip interval. Equations 16 and 18 can therefore be used directly in the causal gate.

C Additional Analysis

C.1 Why identity-metric OBTC is identifiable

Suppose $Ac = \hat{e}$ is feasible and A has full row rank. Every solution is $c = c_{\text{OBTC}} + z$ for $z \in \ker(A)$. The identity-metric OBTC solution in Equation 6 lies in the row space of A and is therefore orthogonal to z. Consequently,

$$\|c\|^2 = \|c_{\text{OBTC}}\|^2 + \|z\|^2. \tag{19}$$

The refresh observation constrains the row-space component but contains no information about z. Setting $z = 0$ is therefore not a separate sample-quality objective; it is the unique choice that introduces no unobserved correction and minimizes the size of neglected higher-order perturbations.

C.2 Causal information flow

At the current retained step, SnapCache has the grid, solver coefficients, stored kept outputs, and their divided differences. These determine every candidate's propagation factors, predicted skip-error scale, and solver budget. The residual becomes available only after the selected skip interval has been traversed. It drives the OBTC correction for that interval, then becomes part of the history used for the next decision. This ordering prevents look-ahead leakage and must be preserved in the sampling implementation.

D Additional Sampling Results

Wan2.2 results in Table 2 cover three compression bands with $\kappa \in \{2.2, 2.8, 7\}$ at 1280×704 resolution. The combined recipe uses 100 VBench prompts, seed 42, and UniPC flow shift 5.

E Sampling Configuration and Reporting

Image trajectories. FLUX.1-dev uses a 50-step Euler sampling trajectory, guidance 3.5, 1024^2 output, and seed 0 for each of the 553 GenEval prompts. Krea-2-Raw uses 52 Euler sampling steps with true classifier-free guidance, giving two transformer calls at each kept trajectory step, with the same resolution, guidance, prompt set, and seed policy. The original Krea settings use $\kappa \in \{1.3, 3.0\}$; the additional gate configuration uses $\kappa \in \{2.0, 4.0\}$. Krea-2-Raw kept steps equal transformer calls divided by two, and compression is $52/\mathrm{Kept} = 104/\mathrm{calls}$. The FLUX table selects operating points from 19 measured configurations, each using 553 prompts and seed 0. Selection minimizes distance to each target compression within each method, rather than selecting by quality.

Video trajectories. Wan2.2-TI2V-5B uses UniPCMultistep with flow shift 5, 50 nominal steps, guidance 5, and seed 42. The 720p study uses 1280×704, 121 frames, and the first 100 VBench prompts. Each step-skipping configuration is paired by prompt and seed with the full trajectory. The final combined results and ablation confirm the resolution, 100 prompts, seed 42, and flow shift 5.

Figure 5 and Table 1 both use a 50-step FLUX reference. The figure's Speedup axis denotes computational compression, not measured wall-clock speedup.

Metric semantics. LPIPS, SSIM, and PSNR are paired sample-fidelity metrics; video values first average across aligned frames. The main tables report paired fidelity and reference FID. FID is computed with torch-fidelity between accelerated and full-step reference image sets. The reported compression is the nominal number of sampling steps divided by mean retained sampling steps and should not be read as measured latency. The image and video aggregates describe variation across prompts rather than repeated-run uncertainty. LIBERO instead reports repeated task episodes.

Action study and statistical reporting. The LIBERO-Long study uses 500 episodes at lower compression and 200 at higher compression. All-branch call compression is $72/\mathrm{Calls}$, giving approximately $2.2\times$ for the first band. Paired McNemar tests compare high-band outcomes.

Compensation configuration. The final recipe enables both buffer compensation and state injection. Near $2.75\times$, LPIPS is 0.0652 without compensation, 0.0606 with buffer-OBS alone, 0.0510 with state injection alone, and 0.0472 with both, corresponding to reductions of 7.1%, 21.8%, and 27.6% relative to no compensation. The ablation holds the gate and tolerance fixed, not the per-prompt retained indices; mean retained-step counts are nearly, rather than exactly, matched. Table 2 uses the updated VBench measurements for the combined recipe.

Few-step distilled model. Table 5 reports Krea-2-Turbo measurements against an 8-step reference, retaining approximately five or exactly three steps. Compression is the reference count divided by the retained count, not measured wall-clock speedup.

Data exclusions. We exclude an uneven FLUX ablation with 280 versus 22 prompts and partial VBench-dimension results covering only 50 of 100 videos.

F KREA-2-RAW AND KREA-2-TURBO QUALITATIVE EXAMPLES

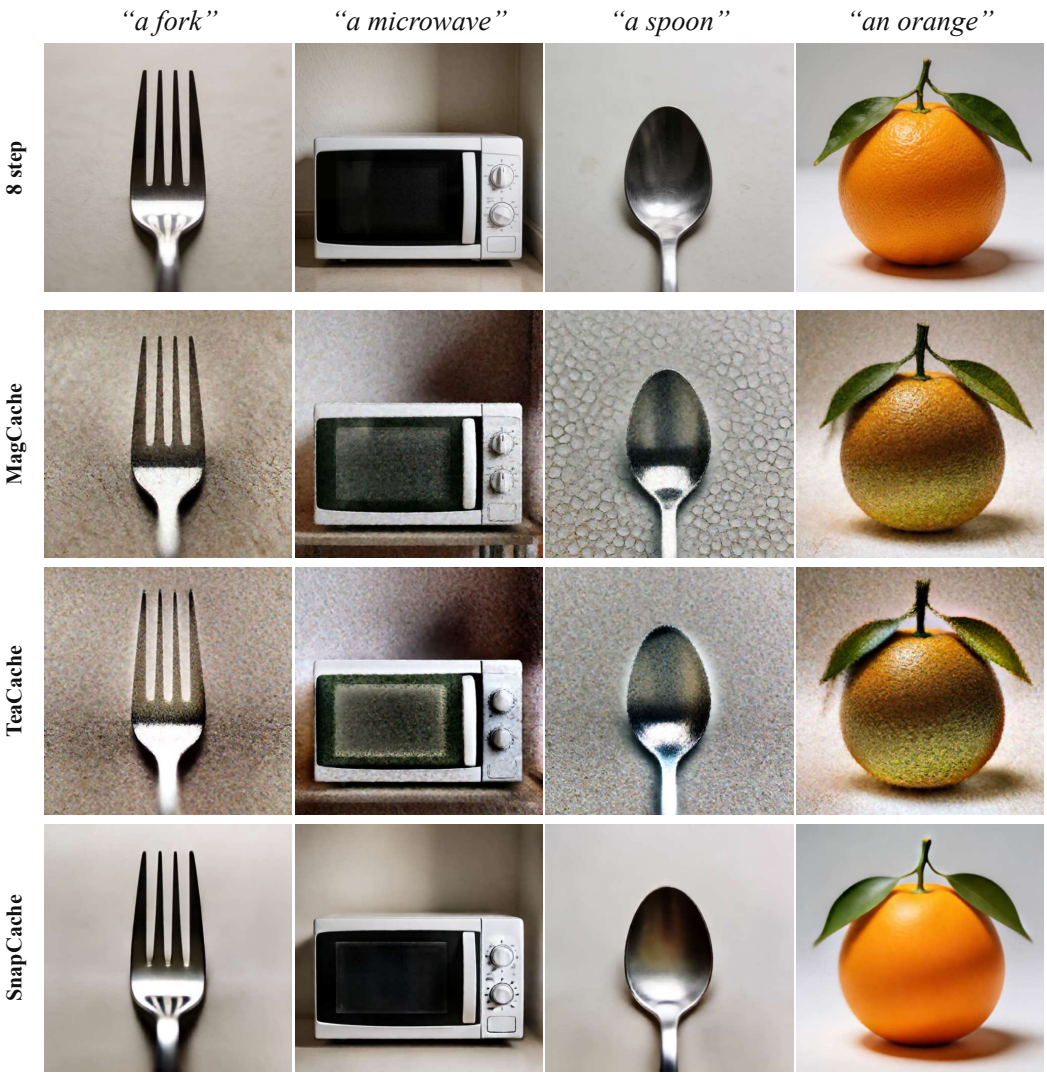

Figure 7. Krea-2-Turbo examples at 2.67× compression, compared with the 8-step reference.

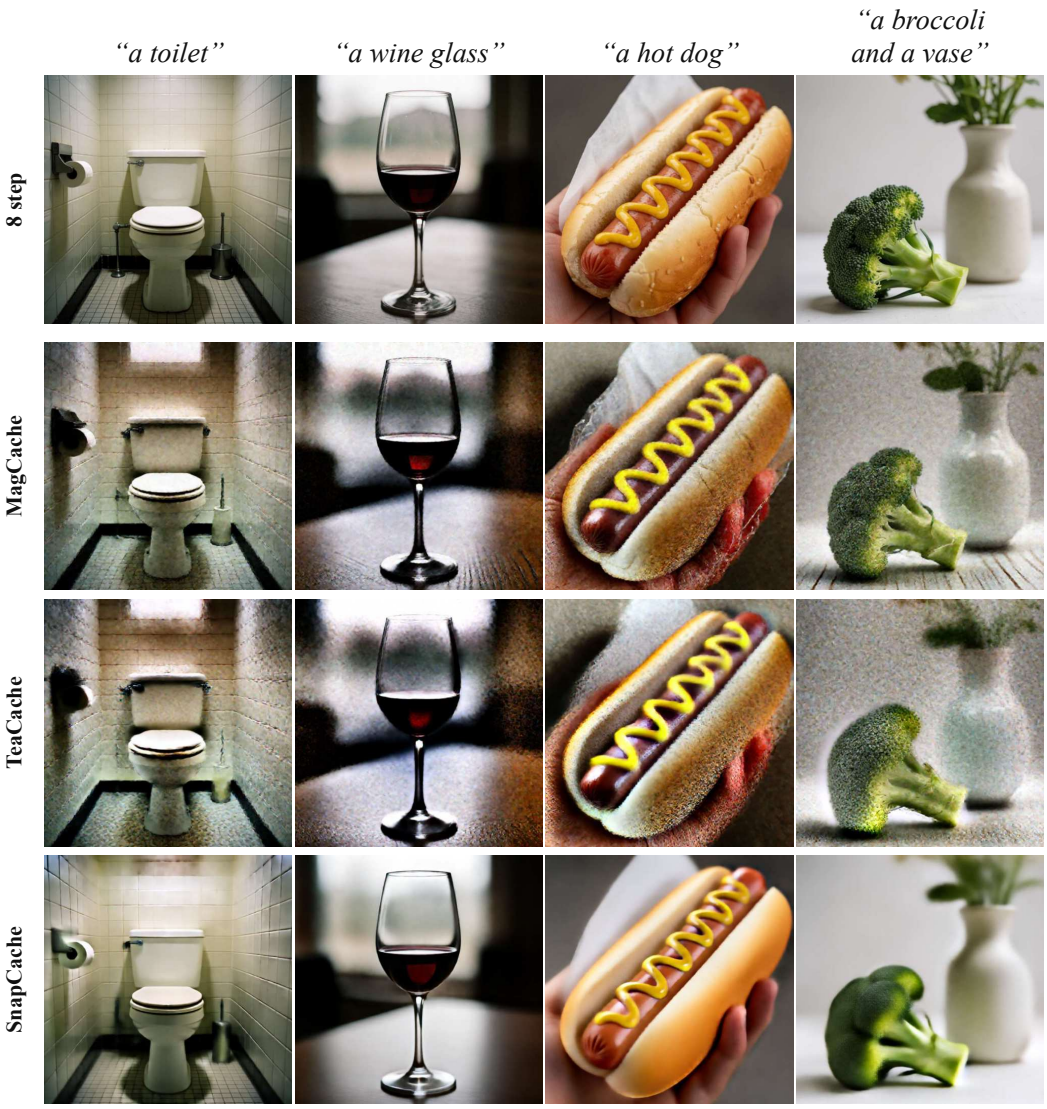

Figure 8. Additional Krea-2-Turbo examples at 2.67× compression against the 8-step reference.



Figure 9. Krea-2-Raw examples at approximately $4\times$ compression, compared with the reference sampler.

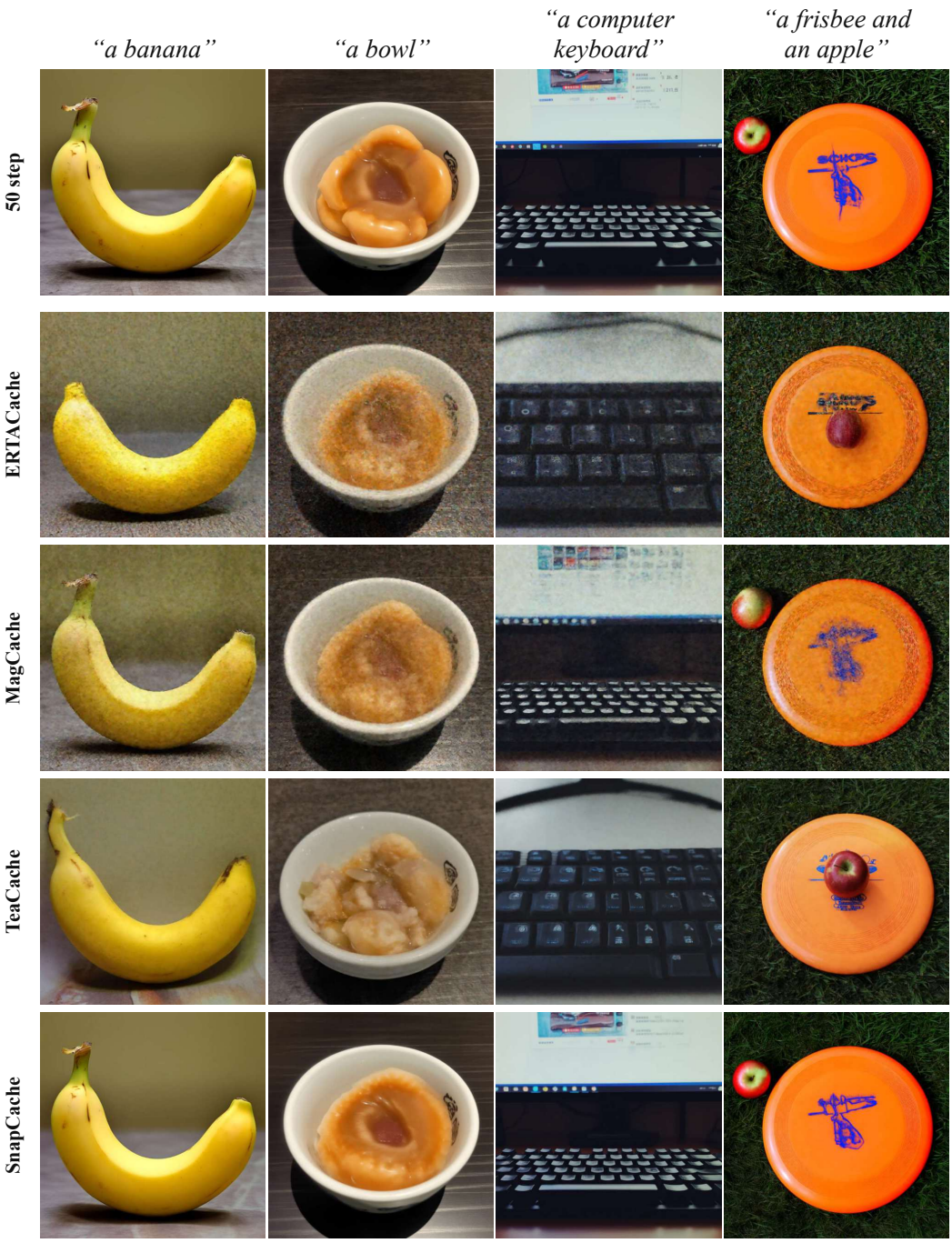

Figure 10. Additional Krea-2-Raw examples at approximately $4\times$ compression against the reference sampler.

G GATE SETTINGS

Table 6. Gate tolerances used in the main tables.

Model	κ	Main table
FLUX.1-dev	2.0, 8.0	Table 1
Krea-2-Raw	1.3, 4.0	Table 1
Wan2.2-TI2V-5B	2.2, 2.8, 7.0	Table 2
Wan2.2 compensation ablation	2.8, 7.0	Table 4
LingBot-VA	1.2, 16.0	Table 3
Krea-2-Turbo	16, 100	Table 5