PROPCACHE: PROPAGATION-CALIBRATED STEP CACHING FOR DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Step caching accelerates diffusion transformers by reusing model outputs, but which steps to reuse determines fidelity at a fixed compute budget. Existing schedulers rely on runtime triggers, candidate replay, or budget-specific search instead of pricing each reuse's terminal impact. We propose PropCache, a training-free compiler built on a propagation price. It factorizes terminal impact into a local defect and a remaining-trajectory gain, then uses one calibration to price every candidate reuse and compile exact-budget schedules for all tiers. Across six flow-matching diffusion transformers spanning 1.3B to 14B parameters, dense and dual-expert backbones, and video and image generation, PropCache establishes the strongest evaluated fidelity–compute point at every backbone's aggressive tier. It lowers Learned Perceptual Image Patch Similarity (LPIPS) by 8–29% against LeMiCa across five video models, leads LeMiCa at all nine HunyuanVideo-13B budgets by 13.5–31.8%, and reaches 4.83× end-to-end speedup. On the full 943-prompt VBench suite, its matched-latency gain is significant at $p = 2 \times 10^{-6}$. Anonymized code is available at `https://anonymous.4open.science/r/propcache-review-5CA2/`.

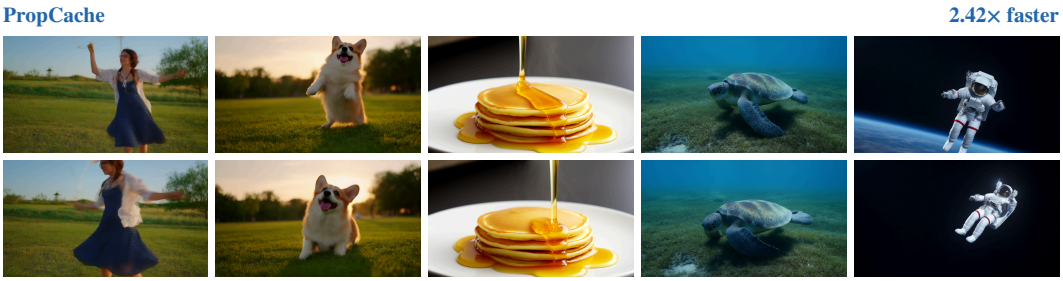

Figure 1: **Five PropCache videos on HunyuanVideo-13B.** Each column shows two frames from one clip; labels give measured end-to-end speedup over dense sampling.

1 INTRODUCTION

Diffusion transformers, or DiTs, generate high-fidelity images and video (Peebles & Xie, 2023; Polyak et al., 2024; Wan Team, 2025), but dozens of sequential model forwards make inference expensive. Step caching saves forwards by reusing earlier outputs (Ma et al., 2024b); the skipped positions determine fidelity at a fixed compute budget. Figure 2 illustrates why.

Existing schedulers trigger reuse from intermediate-state changes (Liu et al., 2025a; Ma et al., 2025; Kahatapitiya et al., 2025; Haghighi & Alahi, 2026; Agrawal et al., 2026) or measure candidate segments and schedules by replay, search, or policy fitting (Gao et al., 2025; Lei et al., 2026; Shao et al., 2026; Cui et al., 2026; Ye et al., 2026). We instead price unmeasured candidate reuses from a shared propagation calibration, avoiding per-candidate replay.

We propose PropCache, a propagation-calibrated schedule compiler. It prices each reuse by combining its local defect with the gain of the remaining denoising trajectory. One calibration per de-

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.

ployment configuration prices all candidate reuses across budgets. An exact bottleneck solver then compiles one mask per budget for use at inference without runtime search.

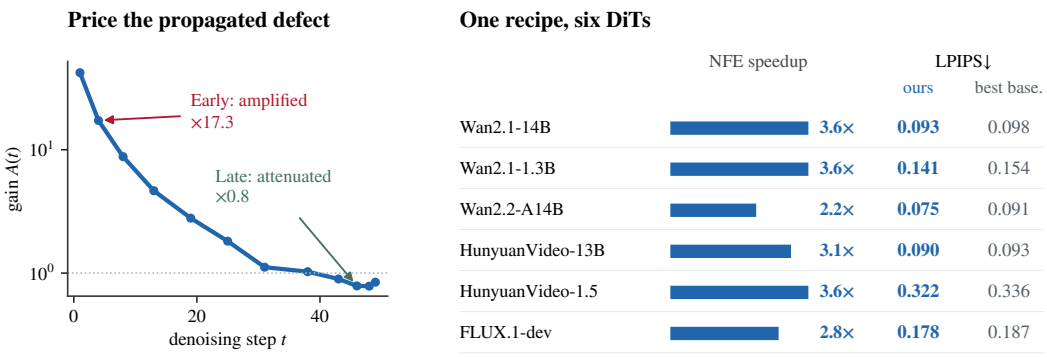

Figure 2: **PropCache at a glance.** Left: a skipped step's final impact depends on propagation gain $A(t)$. Right: the ratio of dense to cached network function evaluations, or NFE, where PropCache matches or improves the best evaluated baseline at the aggressive tier.

Across five video models, PropCache lowers LPIPS by 8–29% relative to LeMiCa (Gao et al., 2025) and reaches $4.83\times$ end-to-end speedup. It also lowers mean LPIPS across all nine HunyuanVideo-13B (Kong et al., 2024) budgets and on the 943-prompt Wan2.1-T2V-14B VBench suite (Huang et al., 2024) at matched latency.

Our contributions are:

- **A propagation price for step caching.** We combine local defects with directly measured propagation gains. Shared cross-prompt gain structure enables reusable pricing without per-candidate replay.
- **A training-free, budget-exact compiler.** We solve the bottleneck objective exactly in $O(n^2k)$ and compile one mask per budget. Inference requires one lookup, with no runtime search or weight updates.
- **State-of-the-art results across six backbones.** PropCache achieves the strongest evaluated aggressive-tier fidelity–compute point on all six, reaches $4.83\times$ end-to-end speedup, and stays within 3×10^{-4} LPIPS of the best method across four shared Wan2.1-14B tiers.

2 Related Work

Trigger-based caching. DeepCache (Ma et al., 2024b), Cache-Me-if-You-Can (Wimbauer et al., 2024), and BWCache (Cui et al., 2025) reuse network features. TeaCache (Liu et al., 2025a), MagCache (Ma et al., 2025), SenCache (Haghighi & Alahi, 2026), AdaCache (Kahatapitiya et al., 2025), ACID (Agrawal et al., 2026), MixCache (Wei et al., 2025), Learning-to-Cache (Ma et al., 2024a), and PAB (Zhao et al., 2024) use local runtime signals to trigger reuse in DiTs.

Budget-constrained scheduling. LeMiCa (Gao et al., 2025) replays reuse-graph edges and solves a lexicographic minimax path; BudCache (Lei et al., 2026) searches schedules per budget. SODA (Shao et al., 2026) and token-wise caching (Zou et al., 2025) use measured sensitivities in dynamic programming, while DPCache (Cui et al., 2026) plans a denoising path. BeCARE (Zhang et al., 2026) allocates refreshes using staleness. Concurrent GCache (Ye et al., 2026) learns propagation weights against a perceptual objective.

Feature correction and forecasting. ERTACache (Peng et al., 2025), IEC (Zhong et al., 2025), FasterCache (Lv et al., 2025), and OTCache (Gao et al., 2026) correct reuse errors; OmniCache (He et al., 2026) exploits finer-grained redundancy. TaylorSeer (Liu et al., 2025b), FORA (Selvaraju et al., 2024), DuCa (Zou et al., 2024), and HyCa (Zheng et al., 2025) forecast intermediate features within retained steps.

Propagation-calibrated pricing. LeMiCa and GCache are closest to our work because both connect reuse to final output quality. LeMiCa averages replayed segment effects across prompts; GCache optimizes propagation weights through a quality objective. PropCache pools directly measured directional gains across prompts, then prices candidate segments without replaying each one. Appendix A compares their calibration and deployment protocols.

3 Problem and Propagation Price Model

3.1 Reference Trajectory, Cache Mask, and the Local Defect

A flow-matching diffusion transformer advances the latent state as

$$x_{t+1} = x_t + \Delta\sigma_t\, v_\theta(x_t, \sigma_t), \qquad t = 0, \dots, n-1, \;\; \Delta\sigma_t < 0. \tag{1}$$

Here $x_t \in \mathbb{R}^D$ is the flattened VAE latent at noise level σ_t, and n is the number of denoising steps. Evaluating v_θ dominates cost, with two backbone forwards under classifier-free guidance. The sampler fixes the noise grid and update; the inference operator fixes guidance and forward execution.

A caching schedule is a binary mask $m \in \{0,1\}^n$. A cached step with $m_t{=}0$ reuses the velocity of its most recent computed **anchor**, $a(t) = \max\{j \le t : m_j{=}1\}$, and injects the **local defect**

$$\xi_t = v_\theta(x_{a(t)}, \sigma_{a(t)}) - v_\theta(x_t, \sigma_t). \tag{2}$$

The initial step is always computed, $m_0{=}1$. The budget $k = \sum_t m_t$ specifies the number of computed steps, with compute fraction k/n. Figure 9 illustrates both operations.

3.2 Remaining-Trajectory Propagation and the Gain Profile

Let $J_t = \partial v_\theta / \partial x$ denote the Jacobian along the reference trajectory. Linearization gives $v_\theta(\tilde{x}, \sigma) - v_\theta(x, \sigma) = J_t(\tilde{x} - x) + O(\|\tilde{x} - x\|^2)$. For a single defect injected at step t followed by computed steps, the terminal deviation e_n is

$$e_n = \Delta\sigma_t\, \Phi(n, t{+}1)\, \xi_t + O(\|\xi_t\|^2), \qquad \Phi(n, s) = \prod_{j=s}^{n-1} \big(I + \Delta\sigma_j J_j\big), \tag{3}$$

where later factors multiply on the left. The propagator Φ transports the injected defect through the remaining trajectory. Its directional norm along cache-defect directions defines the **gain profile** $A(t)$. We measure this profile with one-step probes: cache step t, compute all other steps, and normalize the terminal displacement by the injected defect magnitude.

On the image-generation model FLUX.1-dev, we observe cross-prompt log-gain correlations of 0.94–0.98, with profiles differing by a near-constant scale within $\pm 22\%$. This shared shape supports one pooled profile per deployment configuration across budgets. Global gain rescaling preserves the minimizing schedule by equation 6, making profile shape the key to placement.

Across six backbones, we measure gain-profile max/min ratios of 11–107, with generally larger terminal gains for early defects (Table 18; Figure 3). Appendix J.1 details the probe protocol and per-model analyses.

3.3 Per-Reuse Price

The **reuse price** combines local injection and remaining-trajectory gain to estimate the relative terminal impact of reusing anchor a at step t:

$$\ell(a, t) = A(t)\, |\Delta\sigma_t|\, d(a, t), \tag{4}$$

where $d(a, t)$ is the normalized velocity displacement, or an affine latent-displacement surrogate when velocities are unavailable (Appendix B.1).

We validate the price using independent single-step terminal-error measurements on four backbones spanning an $11\times$ parameter range. Predictions track errors over roughly two decades, with log–log $R^2 = 0.77$–0.86 and slopes 0.72–0.97 (Figure 4; Table 18).

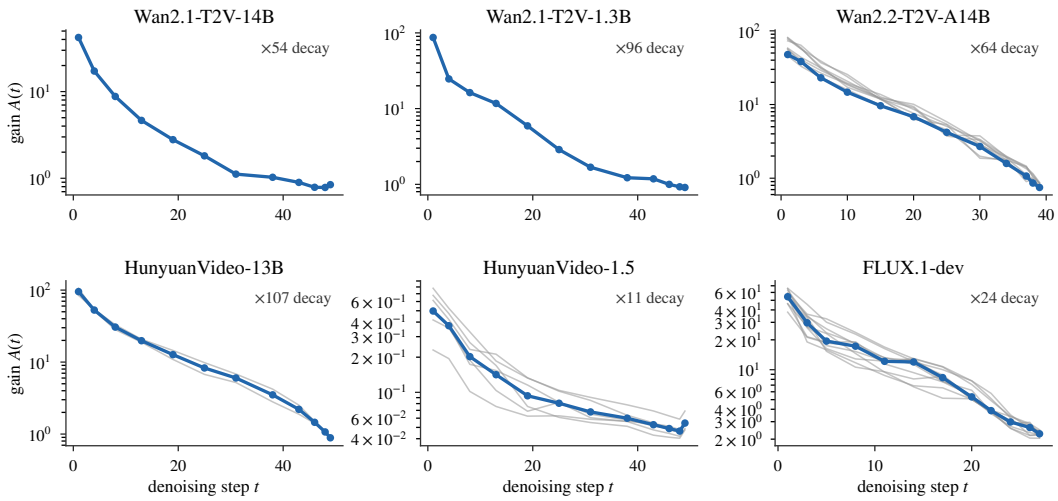

Figure 3: **Propagation gain across six backbones.** Panels use logarithmic vertical axes and report gain scales and decay factors. Blue: pooled profiles; grey: individual calibration profiles available in the archive.

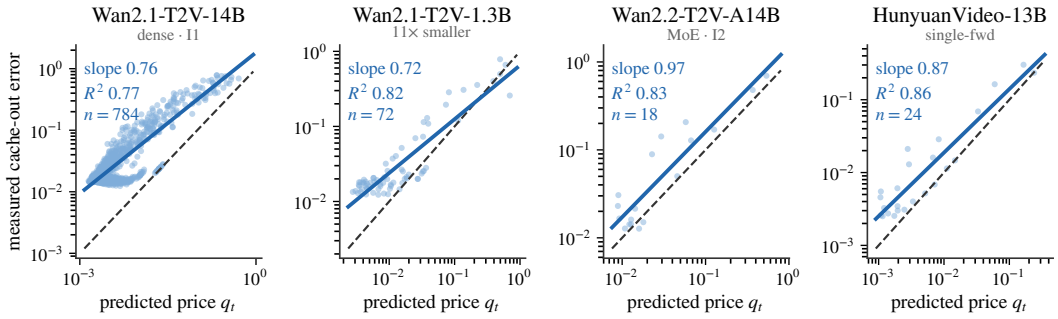

Figure 4: **Price validation across architectures.** Log–log plots compare predicted price q_t with measured single-step cache-out error. Dashed: equality; blue: fitted relation. Panels report slope, R^2, and sample count. I1/I2 denote velocity-/latent-space pricing.

3.4 From One Defect to Many

Single-step probes use fully computed tails, whereas multiple cached steps change both propagation and defect anchors. Although fixed-mask linearized contributions superpose as vectors, their magnitudes need not add. Across 12 two-step probes, we find that summed individual response norms exceed the joint terminal error norm by a median factor of 1.9. We therefore calibrate the aggregation exponent using LPIPS (Section 4.2). Appendices J.2 and J.3 provide the derivations and response decomposition.

4 From Price to Composition and Routing

We aggregate the step prices in equation 4 within each anchor-sharing segment using an ℓ_r norm, then place computed steps to minimize the largest segment cost.

4.1 THE WITHIN-SEGMENT EXPONENT

Let $q(a,t) = \big(\ell(a,i)\big)_{i=a+1}^{t}$ collect the prices of the steps cached against anchor a, and let $L = t-a$ be the segment length. Define

$$S_r(a,t) = \|q(a,t)\|_r = \Big(\sum_{i=a+1}^{t} \ell(a,i)^{\,r}\Big)^{1/r}, \qquad E_r(m) = \max_{\text{segments } (a,t] \text{ of } m} S_r(a,t), \tag{5}$$

where $r \in [1,\infty]$ controls within-segment aggregation. Larger r emphasizes the largest step prices, with $r=\infty$ giving maximum aggregation. We select r using calibration LPIPS over a fixed grid.

4.2 TWO AXES, CALIBRATED PER BUDGET

The compiler jointly selects the exponent and pricing resolution. Pooled pricing uses averaged gains and defects; per-unit pricing uses root-mean-square unit-level prices (Appendix B.1). By positive homogeneity, for any constant $c > 0$,

$$E_r(m;\, cA) = c\, E_r(m;\, A) \qquad \forall m, \tag{6}$$

so global gain rescaling preserves the minimizing schedule. In contrast, the two pricing resolutions can rank schedules differently, motivating joint selection of resolution and exponent. For each budget k, we select on the calibration pool:

$$\big(\dim^*, r^*\big)(k) = \arg\min_{\dim\in\{\text{pool},\text{unit}\},\, r\in\mathcal{R}} \mathbb{E}_{u\in\text{calib}}\big[\mathcal{M}\big(m_{\dim,r}(k)\big)\big], \tag{7}$$

where $\mathcal{R}$ is a fixed grid and $\mathcal{M}$ is LPIPS-alex against a same-process dense reference. All candidates use the same compute budget. Ties favor smaller r, then pooled pricing. The selected mask is shared by all requests at that budget.

Agreement with replay-graph schedules. Like replay-graph scheduling, PropCache minimizes a bottleneck objective, but uses analytical costs. In the ten-tier routing audit, calibration selects three masks identical to the replay baseline and six with lower mean LPIPS (Table 12; Appendix G).

4.3 BUDGET-EXACT COMPILATION AND SERVING

For each exponent, we compute the segment-cost matrix from equation 5 in $O(n^2)$ using row-wise prefix sums, or running maxima for $r=\infty$. An acyclic bottleneck recursion minimizes $E_r(m)$ subject to $|m| = k$ in $O(n^2k)$. Both pricing resolutions share probe records. We evaluate each distinct candidate mask on the calibration pool and store the selected mask per budget. Serving requires one mask lookup, with no runtime search (Algorithm 1; Figure 5).

5 EXPERIMENTS

We evaluate PropCache on six flow-matching diffusion transformers: Wan2.1-T2V-14B and -1.3B, dual-expert Wan2.2-T2V-A14B, HunyuanVideo-13B, HunyuanVideo-1.5, and FLUX.1-dev. Baselines include TeaCache, MagCache, DBCache, LeMiCa, and BudCache. BudCache uses the same reuse runtime as PropCache, isolating schedule placement. LPIPS-alex, peak signal-to-noise ratio (PSNR), and structural similarity (SSIM) measure fidelity to a same-process dense reference. Appendix J.9 specifies cohorts and latency protocols, and Appendix I.1 links results to their source records.

5.1 TEXT-TO-VIDEO

PropCache attains the best LPIPS, PSNR, and SSIM in ten of fourteen compute-matched video groups and stays within 3×10^{-4} of the best LPIPS in two more (Tables 13 and 10). At tight budgets, LPIPS is 8–29% lower than LeMiCa's across five video models, including reductions of 17% on Wan2.2, 22% on HunyuanVideo-13B, and 29% on HunyuanVideo-1.5.

Against BudCache, PropCache reduces LPIPS by 21% on Wan2.2 and 47% on HunyuanVideo-13B at matched budgets. On Wan2.2, the k=16 schedule achieves lower LPIPS and lower measured latency than all six evaluated threshold-based configurations. Figure 18 shows the full latency comparison.

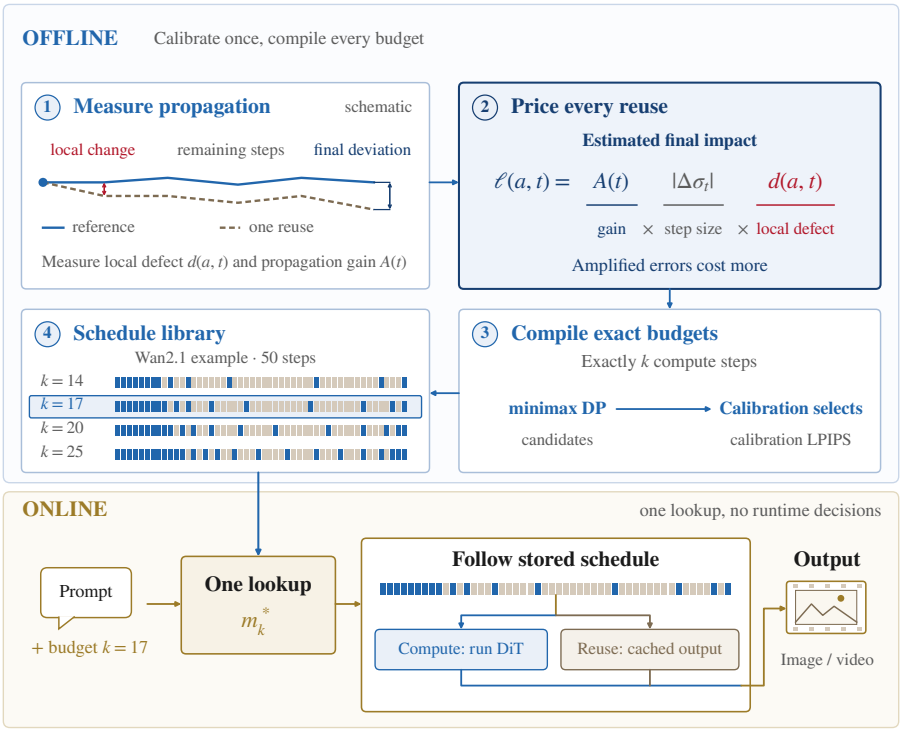

Figure 5: **PropCache from measurement to serving.** Propagation probes price every reuse; exact compilation stores one mask per budget; serving performs one lookup.

Table 1: **Compute-matched text-to-video results.** Bold marks the best fidelity to dense within each group. NFE counts forwards; FLOPs count transformer matrix multiplications across both guidance branches. †: realized compute for triggered methods; ^{e}: end-to-end latency.

		Efficiency			Visual quality		
Method	NFE	FLOPs (P)↓	Latency (s)↓	Speedup↑	LPIPS↓	PSNR↑	SSIM↑
Wan2.2-T2V-A14B^e 16 prompts; dense 40 steps, 134.3 P, 393.4 s							
TeaCache (τ=0.08)†	–	60.4	178.2	2.21×	0.5717	13.1	0.442
BudCache	18	60.4	180.2	2.18×	0.0956	27.2	0.865
LeMiCa	18	60.4	178.5	2.20×	0.0909	27.3	0.873
PropCache	18	60.4	178.5	2.20×	**0.0752**	**28.2**	**0.906**
Wan2.2-T2V-A14B, disjoint split^e 16 unseen units, 24-unit calibration							
BudCache	18	60.4	176.8	2.22×	0.0967	27.6	0.865
LeMiCa	18	60.4	176.9	2.22×	0.0932	27.6	0.867
PropCache	18	60.4	176.9	2.22×	**0.0786**	**28.2**	**0.892**
Wan2.1-T2V-14B^e 16 prompts; dense 50 steps, 167.8 P, 488.3 s; full 943-prompt suite in text							
TeaCache	14	47.0	139.0	3.51×	0.1657	21.2	0.757
LeMiCa	14	47.0	139.0	3.51×	0.1260	22.9	0.804
BudCache	14	47.0	139.0	3.51×	0.1174	23.4	0.814
PropCache	14	47.0	139.0	3.51×	**0.1124**	**24.0**	**0.829**
PropCache	10	33.6	101.2	4.83×	0.1801	21.4	0.746
HunyuanVideo-13B 16 prompts; dense 50 steps, 62.3 P, 93.5 s							
MagCache†	18	22.4	33.7	2.77×	0.0976	28.5	0.889
BudCache	16	19.9	29.8	3.14×	0.1530	25.2	0.860
LeMiCa	16	19.9	29.9	3.12×	0.1027	29.9	0.923
PropCache	16	19.9	29.9	3.13×	**0.0803**	**30.3**	**0.928**

Consistent performance across budget tiers. Across four shared Wan2.1-T2V-14B tiers on the 16-prompt cohort, PropCache stays within 3×10^{-4} LPIPS of the best method (Table 2). The additional $k = 10$ tier achieves $4.83\times$ end-to-end speedup, extending the same calibration to more aggressive deployment. On HunyuanVideo-13B, PropCache lowers mean LPIPS by 13.5–31.8% relative to LeMiCa across all nine budgets from $k = 8$ to 25, with latency matched within 0.3% (Appendix E).

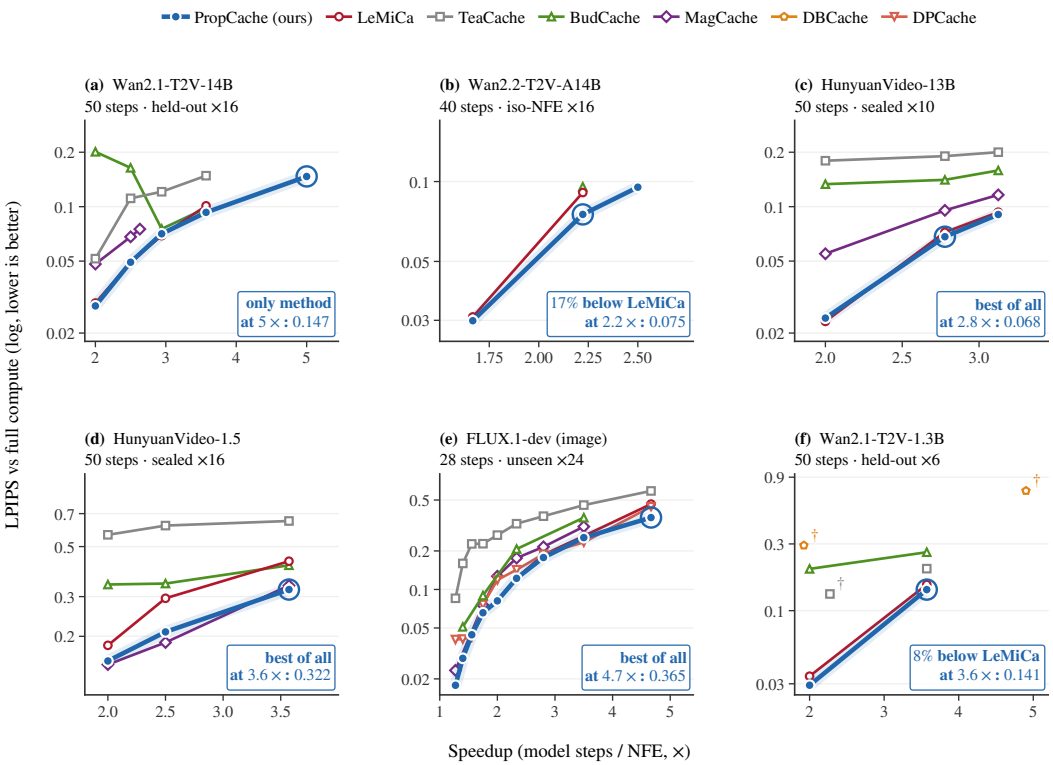

Figure 6: **State-of-the-art aggressive-tier fidelity–compute trade-offs across six backbones.** Speedup is the NFE ratio; † marks realized NFE for a triggered method. Appendix C specifies cohorts and step counts.

Table 2: **One calibration, five exact Wan2.1-T2V-14B budgets.** Bold marks the lowest LPIPS per budget; blanks are unreachable tiers. * covers MagCache's two reachable tiers. Full results appear in Table 13.

Method	LPIPS↓ at budget k of 50 forwards					worst gap to best↓
	10	14	17	20	25	
	4.83×	3.51×	2.90×	2.48×	1.99×	
TeaCache	–	0.1657	0.1337	0.1209	0.0662	0.0657
MagCache	–	–	–	0.0741	0.0549	0.0232*
BudCache	–	0.1174	**0.0783**	0.1612	0.2120	0.1803
LeMiCa	–	0.1260	0.0808	**0.0552**	**0.0317**	0.0136
PropCache	**0.1801**	**0.1124**	0.0784	0.0555	**0.0317**	**0.0003**

Generalization across prompts. On all 943 VBench prompts, Wan2.1-T2V-14B at $k = 14$ lowers mean LPIPS from LeMiCa's 0.0999 to 0.0966, with latency matched within 0.04% and paired-bootstrap $p = 2 \times 10^{-6}$. Figure 17 gives the category breakdown. On Wan2.2, a disjoint split with 24 calibration trajectories and 16 evaluation units gives LPIPS improvements of 16% over LeMiCa and 19% over BudCache at $k = 18$, together with higher PSNR and SSIM. In a separate Wan2.2 VBench evaluation on 88 prompts and two seeds at $k = 18$, PropCache scores higher than LeMiCa on five of six dimensions, with three significant at $p < 10^{-8}$ (Table 8).

Propagation-price validation. On Wan2.2, applying the bottleneck objective to exhaustive single-step reuse measurements yields the same masks as analytical pricing at $k = 18$ and 24 across all 16 units. Price and measured error have pooled Spearman correlation 0.857.

Table 3: **FLUX.1-dev at 1024^2: 88 unseen prompts, two seeds.** CLIP-L measures image–text alignment; HPSv2.1 is Human Preference Score. Latency: median wall time; †: threshold-triggered methods.

		Efficiency			Visual quality				
Method	NFE	FLOPs (P)↓	Latency (s)↓	Speedup↑	LPIPS↓	PSNR↑	SSIM↑	CLIP-L↑	HPSv2.1↑
Dense (28 steps)	28	2.083	2.87	1.00×	–	–	–	–	–
MagCache†	9	0.669	0.95	3.02×	0.2419	20.96	0.7766	0.3566	0.2950
PropCache	10	0.744	1.00	2.87×	0.1696	24.08	0.8330	0.3562	0.2975
Uniform	14	1.041	1.40	2.05×	0.3344	17.17	0.7125	0.3579	0.2942
BudCache	14	1.041	1.40	2.05×	0.1349	25.68	0.8545	0.3562	0.3012
LeMiCa	14	1.041	1.40	2.05×	0.0878	27.41	0.8998	0.3577	0.3008
PropCache	14	1.041	1.40	2.05×	**0.0827**	**28.53**	**0.9077**	0.3572	0.2997
TeaCache†	15	1.116	1.57	1.83×	0.2110	20.65	0.7952	0.3604	0.2996
PropCache	18	1.339	1.80	1.59×	**0.0382**	**32.47**	**0.9514**	0.3582	0.3000



Figure 7: **Qualitative results across models and modalities.** Baselines use their lowest-error evaluated tiers, and PropCache uses its fastest displayed tier. Badges report measured latency and speedup. Figures 15 and 16 provide additional frames and examples.

5.2 Text-to-Image

At 1024^2 on 88 unseen prompts and two seeds, PropCache leads LPIPS, PSNR, and SSIM at $k = 14$, lowering LPIPS by 5.8% relative to LeMiCa at the same 1.40 s latency (Table 3). At $k = 18$, LPIPS falls to 0.0382. CLIP-L and HPSv2.1 remain similar across methods. One shared profile supplies all tiers. Appendix H reports threshold-arm comparisons, and Figure 7 shows image and video examples.

5.3 Ablations and Analysis

Propagation weighting guides compute allocation. We isolate the propagation term $A(t)$ on FLUX by fixing all other components (Table 4). Setting $A(t) = 1$ increases held-out LPIPS by 84% at $k = 14$ and 129% at $k = 18$. On Wan2.1-T2V-14B, uniform propagation weights or prefix-only computation yield 2.2–$5.1\times$ the LPIPS of PropCache (Figure 13).

Table 4: **Propagation-price control on FLUX.1-dev.** Only $A(t)$ varies. *gens* includes probes and candidate-mask evaluation. LPIPS uses 32 held-out prompt–seed units; Δ is relative to measured pricing. Intervals are prompt-clustered 95% bootstrap estimates; brackets for the single-prompt row span its twelve draws. Protocol: Appendix B.6.

	$k = 14$				$k = 18$			
Propagation price	gens	LPIPS	Δ	Interval, %	gens	LPIPS	Δ	Interval, %
$A(t) = 1$	48	0.1883	$+\mathbf{84.0}\%$	$[+59.9, +115.2]$	24	0.1193	$+\mathbf{129.0}\%$	$[+79.8, +194.8]$
$A(t) = (t+1)^{-\gamma}$, searched	1548	0.0940	-8.1%	$[-15.9, -0.5]$	636	0.0490	-5.9%	$[-17.4, +2.6]$
$A(t) = e^{-\lambda t}$, searched	1296	**0.0932**	-9.0%	$[-15.3, -2.5]$	684	**0.0472**	-9.4%	$[-19.7, -0.7]$
$A(t)$ measured, 1 prompt	37	0.0982	-4.1%	$[-9.0, +1.1]$	31	0.0556	$+6.7\%$	$[-3.4, +23.3]$
$A(t)$ measured, 12 prompts, PropCache	156	0.1023	—	—	180	0.0521	—	—

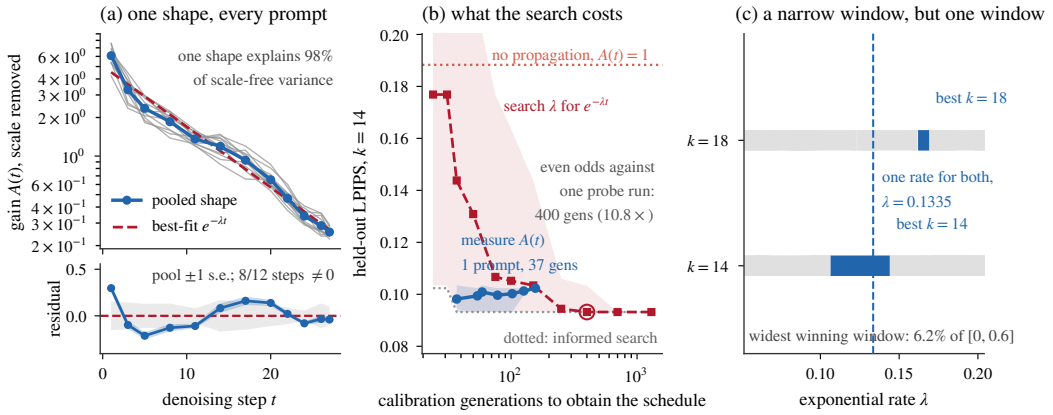

Figure 8: **Shared gain structure and calibration efficiency.** A pooled shape explains 98% of scale-free gain variance. Other panels show held-out LPIPS versus calibration cost and rate intervals selecting the winning schedules.

Complementary pricing and selection. We find that joint selection reduces LPIPS by 18.8–23.9% over pooled pricing with $r = 2$ across four audited FLUX budgets. At $k = 14$, either axis alone changes LPIPS by at most 0.2%, but together they reduce it by 19% (Appendix J.7).

Efficient calibration through direct probes. Single-prompt calibration, including mask evaluation, averages 37 and 31 incremental generations at $k = 14$ and 18. With search capped at 37 generations, the best tested strategy wins only 8–22% against probes in three of four settings (Table 6). A shared twelve-prompt profile plus selection serves both budgets for 192 incremental generations (Figure 8; Table 7). With twelve prompts, four probe steps match full-grid median LPIPS at $k = 14$ and stay within 2.5% at $k = 18$ (Appendix B.6).

6 Conclusion

We introduced PropCache, which turns local cache defects and measured propagation gains into budget-exact schedules. One calibration supports multiple budgets, and inference requires one mask lookup. Across six video and image backbones, PropCache lowers LPIPS by up to 29% relative to LeMiCa and achieves 4.83× end-to-end speedup.

The Use of Large Language Models (LLMs)

LLM-based assistants supported code development and refactoring, manuscript drafting and editing, and mathematical derivations and checks. The authors directed and reviewed this work, verified the derivations and experimental results, and take full responsibility for the paper. Experimental measurements were computed by the evaluation code from generated outputs under the protocol in Appendix J.9; LLM responses were not used as experimental measurements.

REFERENCES

Om Agrawal, Saurabh Agarwal, and Aditya Akella. Acid: Adaptive caching for video generation. *arXiv preprint arXiv:2607.12358*, 2026.

Bowen Cui, Yuanbin Wang, Huajiang Xu, et al. Denoising as path planning: Training-free acceleration of diffusion models with dpcache. In *CVPR*, 2026.

Hanshuai Cui, Zhiqing Tang, Zhifei Xu, Zhi Yao, Wenyi Zeng, and Weijia Jia. Bwcache: Accelerating video diffusion transformers through block-wise caching. *arXiv preprint arXiv:2509.13789*, 2025. ICLR 2026.

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 *NeurIPS*, 2025. Spotlight.

Huanlin Gao, Fang Zhao, Qiang Hui, Fuyuan Shi, Shaoan Zhao, Yantao Li, et al. Otcache: Optimal transport for geometry-aware caching in diffusion models. *arXiv preprint arXiv:2606.31026*, 2026.

Yasaman Haghighi and Alexandre Alahi. Sencache: Accelerating diffusion model inference via sensitivity-aware caching. *arXiv preprint arXiv:2602.24208*, 2026.

Zhaoyuan He, Muhammad Muaz, and Lili Qiu. Omnicache: Multidimensional hierarchical feature caching for diffusion models. *arXiv preprint arXiv:2607.23844*, 2026.

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. In *CVPR*, 2024.

Kumara Kahatapitiya, Haozhe Liu, Sen He, et al. Adaptive caching for faster video generation with diffusion transformers. In *ICCV*, 2025.

Weijie Kong et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Mingkun Lei, Tong Zhao, Liangyu Yuan, and Chi Zhang. Budget-constrained step-level diffusion caching. In *ICML*, 2026.

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 *CVPR*, 2025a.

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*, 2025b.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, et al. Fastercache: Training-free video diffusion model acceleration with high quality. In *ICLR*, 2025.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *NeurIPS*, 2024a.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *CVPR*, 2024b.

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.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Xurui Peng, Chenqian Yan, Hong Liu, Rui Ma, Fangmin Chen, Xing Wang, et al. Ertacache: Error rectification and timesteps adjustment for efficient diffusion. *arXiv preprint arXiv:2508.21091*, 2025.

Adam Polyak, Amit Zohar, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

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.

Tong Shao, Yusen Fu, Guoying Sun, Jingde Kong, Zhuotao Tian, and Jingyong Su. Soda: Sensitivity-oriented dynamic acceleration for diffusion transformer. In *CVPR*, 2026.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Yuanxin Wei, Lansong Diao, Bujiao Chen, Shenggan Cheng, Zhengping Qian, et al. Adaptive hybrid caching for efficient text-to-video diffusion model acceleration. *arXiv preprint arXiv:2508.12691*, 2025.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, et al. Cache me if you can: Accelerating diffusion models through block caching. In *CVPR*, 2024.

Xichen Ye, Yifan Wu, Zhikang Xie, Xiangyu Yue, Cheng Jin, and Weizhong Zhang. From local mismatch to global impact: Optimizing cache reuse policy for efficient diffusion. *arXiv preprint arXiv:2608.13043*, 2026. Concurrent work.

Yuhang Zhang, Junxiang Qiu, Huixia Ben, Zhenhua Tang, Lin Liu, Shuo Wang, et al. Becare: Budgeted cache refresh for diffusion transformer acceleration. *arXiv preprint arXiv:2605.27075*, 2026.

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

Shikang Zheng, Guantao Chen, Qinming Zhou, Yuqi Lin, Lixuan He, Chang Zou, Peiliang Cai, Jiacheng Liu, and Linfeng Zhang. Let features decide their own solvers: Hybrid feature caching for diffusion transformers. *arXiv preprint arXiv:2510.04188*, 2025.

Yunshan Zhong, Weiqi Yan, and Yuxin Zhang. Test-time iterative error correction for efficient diffusion models. *arXiv preprint arXiv:2511.06250*, 2025.

Chang Zou, Shikang Zheng, Evelyn Zhang, Runlin Guo, Haohang Xu, Zhengyi Shi, Conghui He, Xuming Hu, and Linfeng Zhang. Rethinking token-wise feature caching: Accelerating diffusion transformers with dual feature caching. *arXiv preprint arXiv:2412.18911*, 2024.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *ICLR*, 2025.

A Differentiation Matrix

Table 5 compares budget-aware caching systems by cost signal, budget handling and runtime adaptation. Budget-exact methods execute the requested number of full-compute steps by construction. LeMiCa measures its graph once using one replayed trajectory per candidate segment and solves it for each budget. Search- and fit-based methods acquire their signals per budget. The analytical propagation price estimates the terminal impact of a candidate reuse. The sampling schedule is the sequence of noise levels used by the numerical sampler. The deployment operator is the inference operator defined in Section 3.1. It fixes how guidance and the model forward are executed at inference time. PropCache calibrates the cost once per model, sampling schedule and deployment operator.

Method	Cost signal	Budget handling	Runtime adaptation
LeMiCa	replay-measured edge errors	solve per budget, exact	none
BudCache	per-budget search	searched, exact	none
SODA	measured sensitivity tables	additive DP, exact	pruning co-decision
DPCache	path-planning cost tensor	planned, exact	none
OTCache	graph reference + OT warp	interpolated across budgets	none
BeCARE	analytic risk profile	refresh cap, online triggers	prompt-adaptive refresh
GCache	learned perceptual policy	bilevel optimization	tunable reuse policy
PropCache	analytical propagation price	compiled, budget-exact tiers	none

Table 5: Differentiation against budget-aware caching systems; each row was checked against the cited paper's own protocol on 2026-08-18.

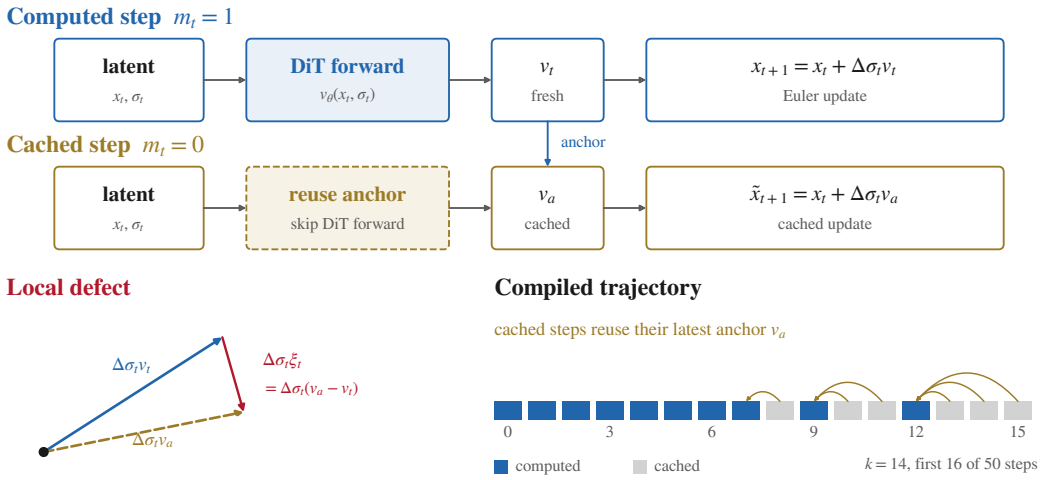

Figure 9: **Step-caching mechanics.** A cached step skips the DiT forward, reuses its latest anchor, and injects the local defect ξ_t.

B Gain-Profile Audits

B.1 Two price instantiations

The price $\ell(a,t) = A(t)\,|\Delta\sigma_t|\,d(a,t)$ of equation 4 uses measured d and A. Two instantiations support the evaluated serving stacks, according to the quantities exposed by the harness.

I1: Velocity substitution. $d = \|v_t - v_a\|/\|x_{\text{end}}\|$ with probe-calibrated gain $A(t) = \beta(t)$. A flow-matching model directly outputs velocity, so I1 uses the sampler output without a fitted constant. It serves the Wan2.1 headline protocol and all cross-model routing.

I2: Affine latent displacement. $d = (\alpha\|x_t - x_a\| + \beta_0)/(\|x_{\text{end}}\| + \epsilon)$ with A calibrated under the deployment operator. This lowest-order surrogate supports external harnesses and is used in the Wan2.2 audit. The external evaluation wrappers used here expose latent states but not intermediate activations. Both instantiations are pooled once per model and held fixed.

Pooled and per-unit pricing. The measured gain in equation 4 supports two pricing granularities:

$$\ell^{\text{pool}}(a,t) = \bar{\beta}(t)\,|\Delta\sigma_t|\,\overline{d(a,t)} \qquad \text{vs.} \qquad \big(\ell^{\text{unit}}(a,t)\big)^2 = \mathbb{E}_u\Big[\big(\beta_u(t)\,|\Delta\sigma_t|\,d_u(a,t)\big)^2\Big], \tag{8}$$

where u indexes calibration units. Pooled pricing uses averaged inputs; per-unit pricing aggregates squared unit-level prices and thus includes cross-unit variation. The measured per-unit-to-pooled price ratio is 0.99–1.22. The two granularities can rank segments differently and select different schedules. Section 4.2 jointly selects the pricing resolution and aggregation exponent.

B.2 THREE EQUIVALENT READINGS OF THE PROPAGATOR

At first order, equation 3 admits variational, linear-response and adjoint interpretations: the state-transition operator of the remaining flow, the endpoint response to a small defect injected at step t, and the costate propagated backward from the terminal loss. The pooled profile absorbs the alignment between the probe direction and the caching direction $v_a - v_t$. Figure 4 audits this alignment through rank validation.

B.3 OPERATOR-ROBUSTNESS CHANNEL DECOMPOSITION

On Wan2.1, responses to defects injected into the conditional branch, the unconditional branch or both agree at the percent level with a single-step cache-out. This agreement supports probing the guidance-combined operator. The guidance-combined operator executes the guided forward as a single call that merges the conditional and unconditional branches. HunyuanVideo uses a single-forward deployment operator, so its probe and deployment operators coincide by construction.

B.4 EARLY-STEP LINEARIZATION AUDIT

At the early steps $t \leq 3$, the linearized price underestimates the measured response by 2–3$\times$.

B.5 SCHEDULE- AND PROBE-DEPENDENCE OF FITTED EXPONENTS: THE V-C AUDIT

The V-C audit probes at $\epsilon = 0.02$ on steps $\{1, 13, 25, 37, 49\}$ under shift-3, shift-5 and uniform schedules, each evaluated against its own all-compute reference. Fitted decay exponents span 0.32–0.75, compared with 0.96 ± 0.25, and $\beta(1)/\beta(49)$ rescales opposite to the pure $(1 - \sigma)$ prediction. The probes are partially saturated at this amplitude. These measurements constrain exponent transfer across schedules. The ratios also vary across schedules, contradicting a step-count-only model. The placement layer therefore uses the measured curve and its rank; the parametric form summarizes that curve.

B.6 PROTOCOL FOR THE PROPAGATION-PRICE CONTROL

Table 4, Table 6, Table 7 and Figure 8 report a FLUX.1-dev campaign at two budgets. The control isolates the propagation term at pooled pricing resolution; the deployed FLUX tier uses per-unit pricing at k=14.

Shared protocol. The solver, exponent grid, pooled pricing resolution, budget, twelve calibration prompts, selection rule and LPIPS readout are identical across arms. Only the propagation term $A(t)$ of equation 4 changes. The evaluated grid is $p = r/2 \in \{1, 1.1, \ldots, 2, 2.5, 3, 4, 6, \infty\}$. The ten audited tiers select $p \in [1, 2.5]$. The stored price table contains $q_t(a)^2$. Dividing it by the measured $A(t)^2$ and multiplying by a candidate $A(t)^2$ compiles every arm from the same frozen defect measurements. Restoring the measured gain reproduces all 64 deployed pooled masks exactly.

Price families. The four families are $A(t) = 1$; $A(t) = e^{-\lambda t}$ with λ on 1201 points over $[0, 0.6]$; $A(t) = (t{+}1)^{-\gamma}$ with γ on 1201 points over $[0, 6]$; and the measured gain from probe pools of 1 to 12 prompts. Each rate is crossed with the full exponent grid. Every distinct compiled schedule is generated on all twelve calibration prompts, and each family selects its calibration argmin. At k=14, the exponential family produces 108 candidate schedules and the full-pool measured price produces 1. λ=0 and γ=0 both reproduce $A(t) = 1$, so each searched family includes the no-propagation price.

Generation accounting. The *gens* column charges one generation per probe and one generation per distinct compiled schedule per calibration prompt. A probe is a perturbed replay of an existing trajectory and is charged as a full generation. The ledger measures incremental price selection costs; all arms share the frozen local-defect table and dense references. Across both budgets, one full probe pool and schedule selection cost $144 + 12 + 36 = 192$ generations: 144 probes, 12 schedule evaluations at k=14 and 36 at k=18. The k=18-only total is 180.

Table 6: **Rate-search cost and held-out fidelity.** Each method is charged one generation per calibration prompt per distinct compiled schedule. Search uses only purchased calibration scores; LPIPS is evaluated on 32 held-out units. Search has a hard cap of 37 generations; single-prompt probe costs average 37 at k=14 and 31 at k=18, with variation across draws. Win rates compare individual search and probe runs over their empirical product distribution, using 2000 search restarts. The single-prompt comparator is fixed across search caps; even odds marks the first tested cap reaching a 50% win rate. The final row is an oracle reference given the optimal rate neighbourhood in advance.

	$A(t) = e^{-\lambda t}$		$A(t) = (t{+}1)^{-\gamma}$	
Rate search strategy	k=14	k=18	k=14	k=18
win rate against the probe, search capped at 37 *generations*				
random	0.09	0.22	0.08	0.40
log-random	0.06	0.18	0.04	0.23
coarse-to-fine	0.00	0.02	0.00	0.68
search cap at even odds for the best of the three strategies				
even odds at	400	150	400	37
calibration-selected optimum after full-grid evaluation				
held-out LPIPS	0.0932	0.0472	0.0940	0.0490
generations	1296	684	1548	636
against the measured price	−9.0%	−9.4%	−8.1%	−5.9%
oracle search with known rate neighbourhood, capped at 37 *generations*				
known neighbourhood	0.46	0.95	0.53	0.99

Budget-capped search replay. Table 6 replays rate searches from the enumerated schedules. A two-point uniform grid contains the endpoints, and a three-point grid adds the midpoint. At k=14, a six-point grid already matches the full grid's held-out LPIPS. The full sweep's 1296 generations record the cost of that enumeration.

Each search proposes rates sequentially, snaps them to the enumerated grid and charges twelve generations for each newly encountered schedule. The run stops when the next proposal would exceed its budget. The search budget is an upper limit on expenditure. Selection uses the calibration means of evaluated schedules, and the selected schedule is assessed on the held-out cohort. The three search strategies are uniform random over the swept range, log-uniform over $[\text{hi}/600, \text{hi}]$, and coarse-to-fine. The last strategy starts with three jittered points and recursively searches a half-width bracket around the current best. An oracle-neighbourhood control samples $\mathcal{N}(\text{rate}^\star, 0.02)$ with the optimal rate supplied in advance. Each strategy uses 2000 restarts per budget.

Win rates compare one search run with one single-prompt probe run over the product of their distributions. The probe samples one of the twelve calibration prompts; its mean costs across those twelve choices are approximately 37 generations at k=14 and 31 at k=18, with expenditure varying by prompt. The search comparison uses a 37-generation cap. Against the same single-prompt probe distribution at every search budget, the best of the three exponential-rate strategies first exceeds a 50% win rate at tested budget points of 400 and 150 generations for k=14 and k=18, respectively. The corresponding tested points for the power law are 400 and 37 generations.

Rate windows and candidate counts. Figure 8c shows the schedules compiled by the 1201-point rate sweep. At k=14, the exponential family produces 108 distinct schedules, and the winning schedule occupies 6.2% of the swept range. Its winning window at k=18 is 1.2%. The full-grid exponential search achieves the lowest held-out LPIPS in the control; budget-capped searches depend on locating these windows.

Offline enumeration of the archived price tables uses the same 1201-point rate sweep per family. The exponential family produces 26–138 distinct schedules per budget on FLUX.1-dev at 28 steps, 441–701 on HunyuanVideo-13B and 770–1059 on HunyuanVideo-1.5, both at 50 steps. The power law gives counts of the same order. These are candidate counts for the stated finite grids. Probe generation counts are determined by the chosen prompt and step grids; schedule evaluation and per-generation runtime also contribute to calibration cost.

Table 7: **Rate transfer across budgets on FLUX.1-dev.** Selection uses calibration scores; evaluation uses held-out LPIPS. Bracketed percentages are relative to each budget's independently searched grid. Transferring a frozen rate increases LPIPS more at the lower budget. A jointly selected rate stays within 3.8% of both budget-specific optima despite their narrow argmin intervals. A shared rate supports budget-specific schedules; obtaining it still incurs the grid-search costs listed below.

	$A(t) = e^{-\lambda t}$		$A(t) = (t{+}1)^{-\gamma}$	
Rate obtained by	k=14	k=18	k=14	k=18
held-out LPIPS				
this budget's own grid	0.0932 (+0.0)	0.0472 (+0.0)	0.0940 (+0.0)	0.0490 (+0.0)
the *other* budget's grid, frozen	0.1034 (+11.0)	0.0490 (+3.8)	0.1172 (+24.7)	0.0502 (+2.4)
both grids, one shared rate	0.0932 (+0.0)	0.0490 (+3.8)	0.0940 (+0.0)	0.0502 (+2.4)
shared probe, PropCache	0.1023 (+9.8)	0.0521 (+10.3)	0.1023 (+8.9)	0.0521 (+6.3)
no propagation, $A(t) = 1$	0.1883 (+102.1)	0.1193 (+152.7)	0.1883 (+100.3)	0.1193 (+143.3)
generations charged to obtain the rate				
this budget's own grid	1296	684	1548	636
the other budget's grid	684	1296	636	1548
both grids	1980	1980	2184	2184
shared probe, PropCache	156	180	156	180

Cross-budget transfer. Figure 8c shows disjoint λ intervals for the two budgets' exact winning schedules. Table 7 measures near-optimal transfer across the $19{,}216$ rate–exponent configurations enumerated at both budgets. Freezing a rate tuned at k=14 increases held-out LPIPS at k=18 by 3.8% for the exponential and 2.4% for the power law relative to per-budget search. The reverse transfer increases LPIPS by 11.0% and 24.7%. Joint selection minimizes worst-case relative calibration regret. The resulting exponential configuration, λ=0.1335 at p=1.2, stays within 3.8% of per-budget search at both budgets. Thus, one searched configuration can serve both budgets. The executed full-grid enumerations cost 684 and 1296 generations at k=18 and k=14, respectively. These costs describe the performed sweeps; the sparse-grid and budget-capped results above measure lower-cost routes to selecting a rate. A full shared probe pool plus selection at both budgets costs 192 generations.

Cohort and coverage. Calibration uses six DPG and six GenEval prompts. The held-out cohort contains eight DPG and eight GenEval prompts at two seeds each. It is disjoint from both calibration and the paper's other verification cohort, with the same prompt-family balance as calibration. Every candidate schedule is generated and scored on every held-out unit, providing complete records for calibration-selected results, the held-out skyline and search replay. A second campaign generates all additional schedules compiled by sparse probe subgrids, giving full coverage at every subgrid density. Regenerating the deployed campaign's arms in this harness reproduces their LPIPS with a mean absolute difference of 2.5×10^{-4}, or 0.46% of the arm-to-arm spread.

Probe-pool size. The full measured price uses twelve calibration prompts probed at twelve grid steps. The offline subgrid audit enumerates all 4082 subsets containing two to eleven of those steps, pools over all twelve prompts and reports the held-out LPIPS of each selected schedule. Inclusion requires generated results for every schedule a subgrid compiles; the second campaign supplies complete coverage. Four steps match the full pool's held-out LPIPS at k=14, 0.1024 against 0.1024. At k=18, they yield 0.0534 against 0.0521, an increase of about 2.5%; six steps yield an increase of about 2.4%.

One prompt probed at six steps uses six probes. Including evaluation of the distinct compiled schedules, its mean total cost is 38.71 generations at k=14 and 28.18 at k=18. Median held-out LPIPS is 0.1006 at k=14, below the full-pool value, and 0.0544 at k=18, within 4.4% of the full pool. The full-pool value lies inside the run-to-run band. At k=18, the single-prompt six-point probe matches or beats the full pool in 32% of draws. The twelve-prompt pool primarily reduces variability.

Finite-grid mask agreement. Offline re-pricing of the frozen tables compares measured-gain schedules with decay-family schedules on each archived backbone. At every deployed budget, λ and γ are each swept over 1201 points and crossed with the full exponent grid:

Backbone	steps	budgets	measured schedules reachable by a decay
FLUX.1-dev	28	4	2/9
HunyuanVideo-13B	50	9	37/79
HunyuanVideo-1.5	50	4	8/50
total		17	47/138

Across the stated finite grids, 91 of the 138 measured-gain schedules, or 66%, have no matching decay-family schedule. The narrowest observed rate window spans one grid step. These counts measure mask agreement at the enumerated resolution. Table 4 separately measures quality and shows that searched decay schedules can achieve lower LPIPS. Figure 8a shows reproducible cross-prompt residuals from the best-fit exponential.

Scale invariance. Rescaling the measured gain by $c \in [10^{-6}, 10^{6}]$ leaves the schedule unchanged in $512/512$ compilations across budgets and exponents, numerically verifying equation 6. This result applies to a shared global scale. Per-unit scales remain inside the aggregation and are evaluated through the pricing-resolution audit in Section 4.1.

Calibration efficiency. The price control links propagation weighting to effective placement and measures the cost of obtaining each schedule. Small probe pools produce competitive schedules with few measurements. Pooling twelve prompts resolves the deviation from an exponential profile at a 3.3:1 structure-to-noise ratio, compared with 1.0:1 for a single prompt. The subgrid and search comparisons quantify how this information translates into schedule quality and calibration cost.

B.7 Per-model rank validation panels

Figure 10 shows the single-step price validation on the two additional model families, using the same gates fixed before generation, $R^2 \geq 0.70$ and slope in $[0.6, 1.4]$.

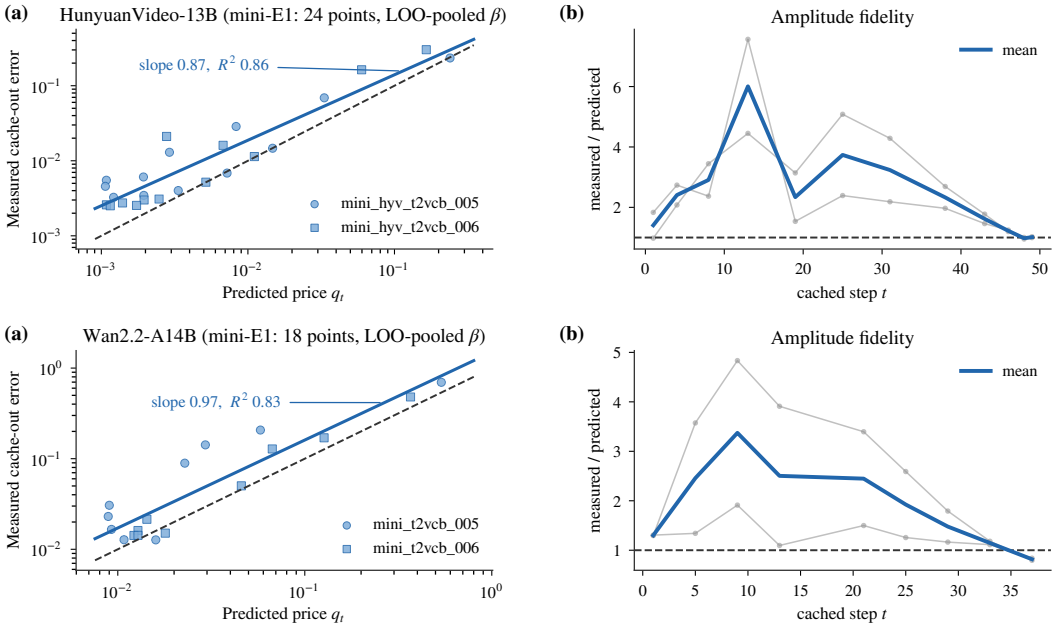

Figure 10: **Single-step price validation on the two additional model families.** Log–log rank plots with LOO-pooled β, amplitude-ratio panels at right. *Top*: HunyuanVideo-13B, n=24, R^2=0.865, slope 0.873. *Bottom*: Wan2.2-T2V-A14B, n=18 after excluding the forced-computed expert-boundary step 17, R^2=0.834, slope 0.970.



Figure 11: **Images generated on FLUX.1-dev with PropCache at** $2.87\times$**,** 10 **of** 28 **forward passes.** Five of the 88 unseen prompts $\times\,2$ seeds of Table 3, chosen for content variety before any pixel was viewed; their ranks in the arm's LPIPS distribution are 51, 67, 87, 153 and 156 of 176, so the set is not skewed towards the units that track the dense sampler most closely. Tiles are PropCache outputs only and make no fidelity claim; the scored comparison is Table 3.

C Supporting Figures

Figure 12 reports the coherence regime of all six models. Figures 13 and 14 examine propagation weighting and schedule granularity on Wan2.1-T2V-14B. The frontier figure's panel-level cohorts are listed below.

Cohort provenance for Figure 6. Panel (a) uses Wan2.1-T2V-14B, 50 steps and held-out $\times16$; (b) uses Wan2.2-T2V-A14B, 40 steps and iso-NFE $\times16$; (c) uses HunyuanVideo-13B, 50 steps and held-out $\times10$, disjoint from the 16-unit iso-latency audit of Table 9; (d) uses HunyuanVideo-1.5, 50 steps and held-out $\times16$; (e) uses FLUX.1-dev, 28 steps and the 24-unit unseen pilot cohort, disjoint from the 88×2 scaleup split of Table 3; (f) uses Wan2.1-T2V-1.3B, 50 steps and held-out $\times6$ at the two budget-exact tiers of Table 10. LeMiCa follows its official whole-step semantics with $\mathrm{fn}{=}0$. These panels compare PropCache with external methods; Table 12 reports within-family routing comparisons.

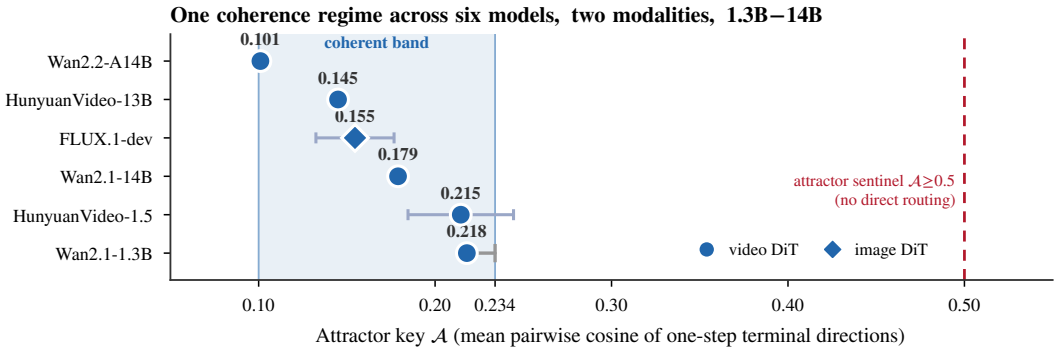

Figure 12: **Coherence across six models.** The attractor key $\mathcal{A}$ of equation 16 is measured on a fixed 12-point probe grid. All six models lie in the shaded coherent band $[0.10, 0.234]$, below the dashed attractor sentinel $\mathcal{A}\geq0.5$ that disables direct routing. Wan2.1-T2V-1.3B shows a solver whisker; HunyuanVideo-1.5 and FLUX show ±1 s.d. over probe units.

D Quality Readout and Per-Category Breakdown

This section extends Section 5.1 with category-level results for the 943-prompt Wan2.1-T2V-14B suite and VBench measurements of generation quality. The overall per-prompt win rate against LeMiCa is 55.2%; category win rates reach 82% for human action and 71% for multiple objects. VBench dimensions are reported separately to preserve their scales: imaging quality uses 0–100, and the remaining dimensions use 0–1. These measurements complement the fidelity results in Table 1. In Table 8, both caches are within 2.4% of the dense arm on all six dimensions, alongside a 17% LPIPS gap between the caches at the same operating point.

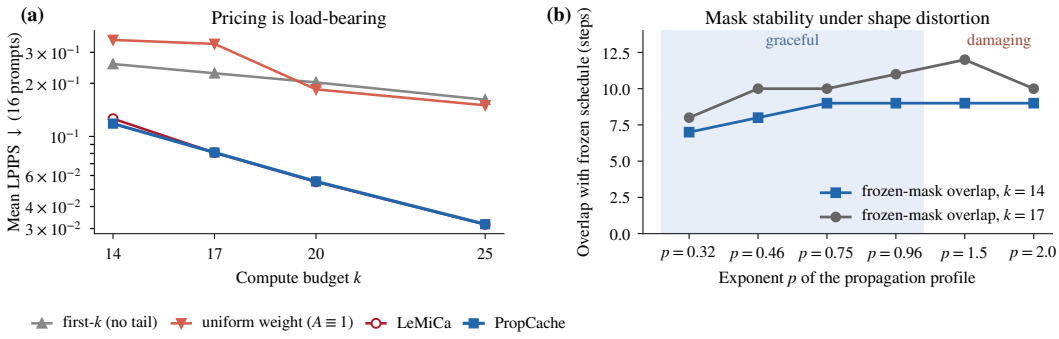

Figure 13: **Propagation weighting and profile-shape sensitivity.** Wan2.1-T2V-14B, 16 held-out prompts. (a) Mean LPIPS for uniform A and for schedules without tail compute, compared with PropCache and LeMiCa at each budget. (b) Sensitivity of the compiled schedule to profile-shape distortion p.

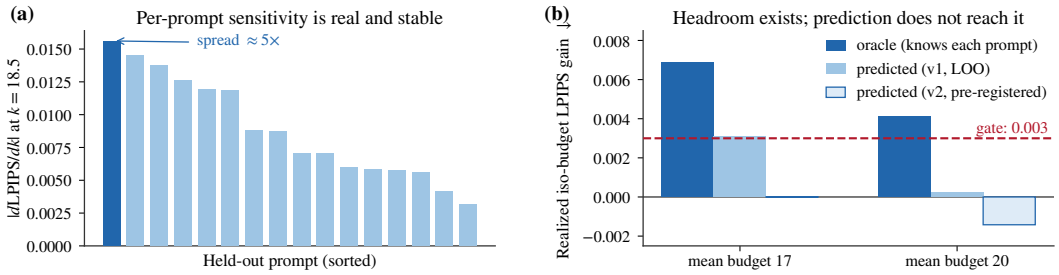

Figure 14: **Pooled and per-prompt schedules.** Wan2.1-T2V-14B, 16 held-out prompts. (a) Per-prompt budget sensitivity is stable at k=18.5. (b) A schedule reconstructed from propagation-side features matches the pooled schedule. The per-prompt oracle alone exceeds the gain threshold fixed before evaluation.

Table 8: **VBench generation quality on Wan2.2-T2V-A14B.** All three methods use one process and 88 prompts $\times$ 2 seeds. PropCache and LeMiCa use exactly k=18 forward passes at matched wall time. Scores are raw VBench outputs; comparisons use paired per-clip differences. p is from a two-sided exact sign test. **Bold** marks the better caching method.

VBench dimension ↑	Dense	LeMiCa	PropCache	Δ vs. LeMiCa	Win/loss	p
Subject consistency	0.9522	0.9481	**0.9514**	+0.0034	152/24	$<10^{-23}$
Motion smoothness	0.9796	0.9793	**0.9796**	+0.0003	136/40	$<10^{-12}$
Temporal flickering	0.9597	0.9615	**0.9616**	+0.0001	91/85	0.71
Aesthetic quality	0.6321	0.6192	**0.6257**	+0.0066	127/49	$<10^{-8}$
Imaging quality	70.40	69.26	**69.53**	+0.27	100/76	0.083
Dynamic degree	0.739	**0.739**	0.722	−0.017	0/3	0.25

Wan2.2 generation quality. LPIPS measures fidelity to the dense reference. VBench (Huang et al., 2024) assesses the dense arm and both budget-exact k=18 arms on subject consistency, motion smoothness, temporal flickering, aesthetic quality, imaging quality and dynamic degree over 88 prompts $\times$ 2 seeds. At matched budget, runtime and wall clock, PropCache scores above LeMiCa on five of six dimensions in Table 8. Per-clip sign tests reject parity on subject consistency, motion smoothness and aesthetic quality at $p < 10^{-8}$. Subject consistency closes 81% of LeMiCa's gap to dense, with deviations of -0.0008 and -0.0042, respectively. Motion smoothness is within 6×10^{-5} of dense for PropCache, while LeMiCa is 2.7×10^{-4} below dense. These two dimensions agree with the price model's predicted advantage. Dynamic degree is the exception to the five-dimension lead; this binary motion flag is tied on 173 of 176 pairs. Temporal flickering scores are 0.9616 and 0.9615 for the two caches, both above the dense score of 0.9597. Thus, the lower LPIPS accompanies

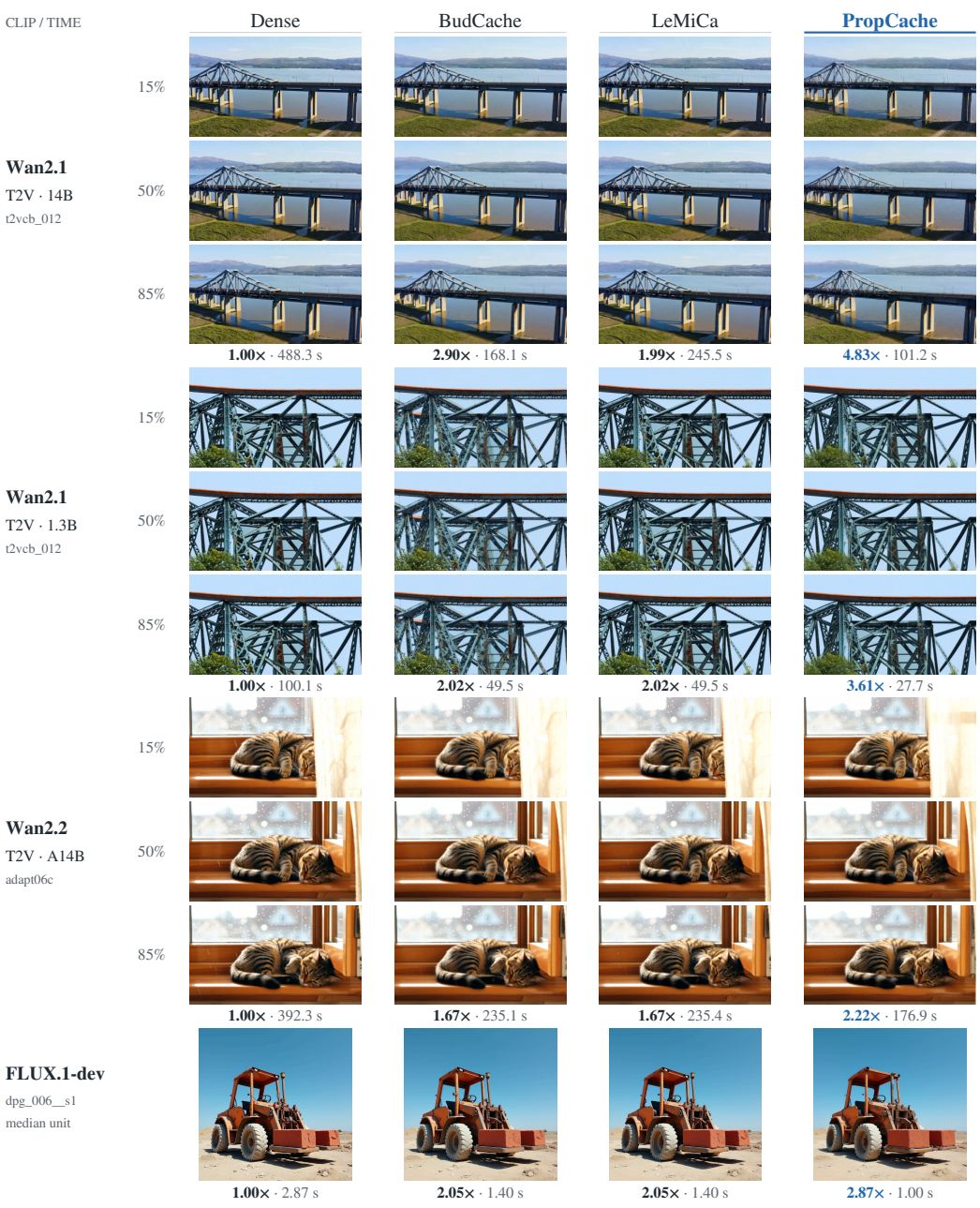

Figure 15: **Three instants per clip, one schedule.** The grid of Figure 7 at 15, 50 and 85% of each clip. Frame times are fixed across methods and selected independently of caching. Each cell contains three frames from one clip at a single operating point.

higher scores on five VBench dimensions and closer agreement with dense on subject consistency and motion smoothness.

E Full HunyuanVideo-13B Iso-Latency Sweep

Table 14 expands the HunyuanVideo-13B summary curves in Section 5.1. For each budget, it reports end-to-end and forward-pass speedups, LPIPS, SSIM and PSNR against the dense reference, and iso-wall LPIPS reductions against MagCache. Across 112 reachable cells, the LPIPS

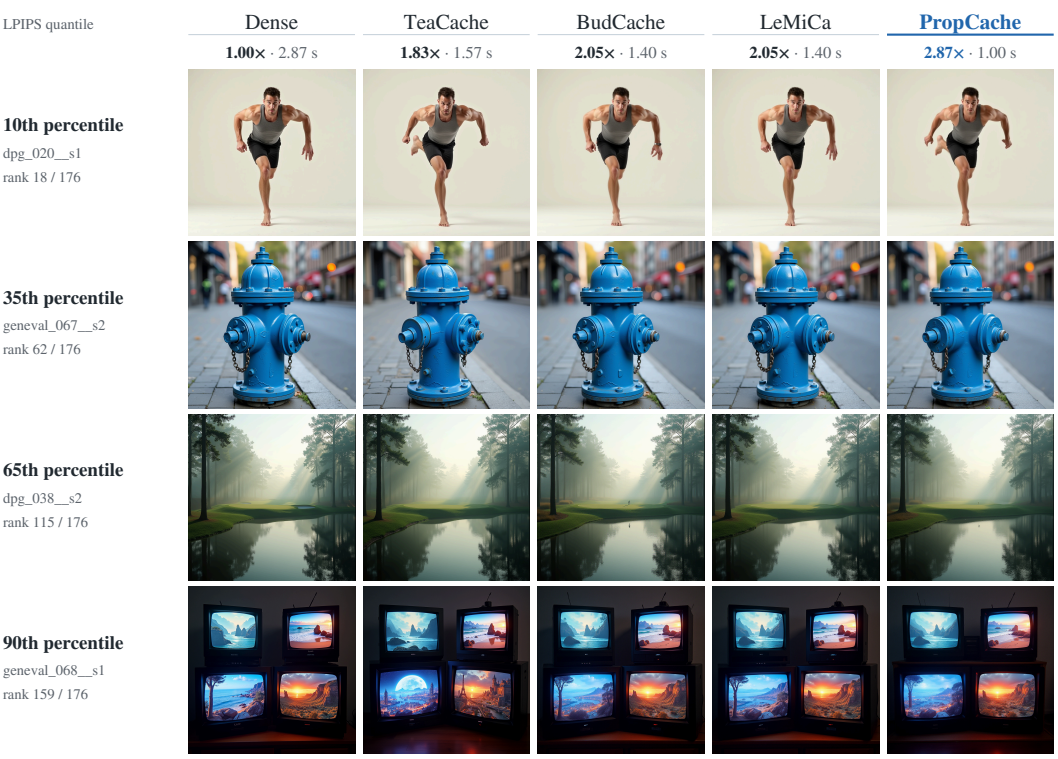

Figure 16: **FLUX.1-dev across the error distribution.** Four unseen units at the 10th, 35th, 65th and 90th percentiles of PropCache LPIPS, each rendered at native 1024^2 resolution. TeaCache at $1.83\times$ alters content preserved by PropCache at $2.87\times$.

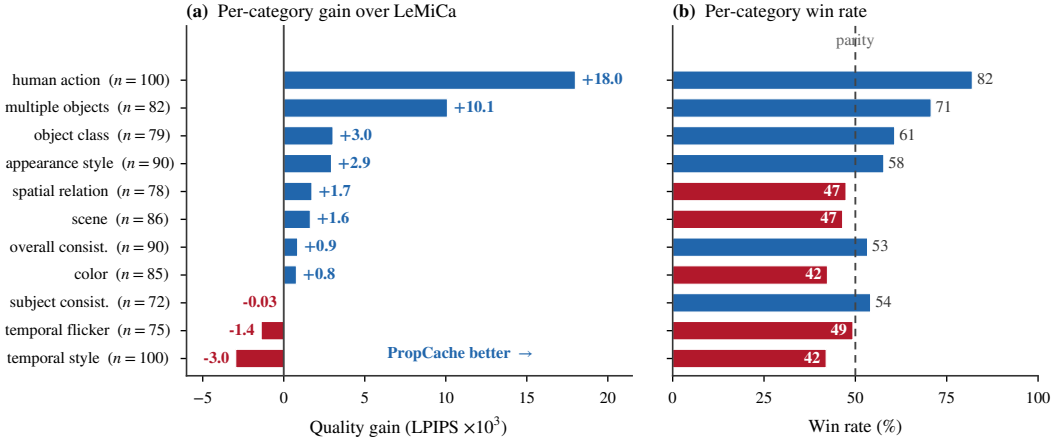

Figure 17: **Category-level fidelity and win rates.** Wan2.1-T2V-14B at k=14 against LeMiCa across all 11 VBench categories, using the 937-prompt per-category subset of the 943-prompt suite. (a) LeMiCa LPIPS minus PropCache LPIPS in units of 10^{-3}; rightward blue bars indicate lower error for PropCache. (b) Per-category win rates against the 50% parity line. Categories are ordered by effect size.

reduction has a mean of 39.7% and a median of 41.4%. Table 15 gives all 144 per-cell measurements and comparison outcomes. Both tables are generated from the archived audit by `scripts/make_appendix_tables.py`.

Table 9 summarizes all three baselines. The audit contains 432 iso-latency cells from 16 units × 9 budgets × 3 baselines. PropCache is better in 350, tied in 12 and worse in 4; the remaining 66 cells have operating points outside the baselines' attainable latency ranges.

Baseline	Cells	PropCache better	Tie	Baseline better	Unreachable
MagCache, online trigger	144	98	11	3	32
TeaCache, online trigger	144	125	0	0	19
BudCache, per-budget search	144	127	1	1	15
Total	432	**350**	12	4	66

Table 9: **HunyuanVideo-13B iso-latency audit.** 16 units × 9 budgets × 3 baselines. Each cell evaluates a baseline's end-to-end wall-time–quality curve at PropCache's wall time, with a ±0.005 LPIPS tie band. Unreachable operating points are excluded from win/tie/loss counts. Table 15 details the MagCache comparison.

Generation-quality protocol and results. The VBench audit uses 200 paired units, including 176 unseen units, with three arms producing 600 videos scored on 6 dimensions. Gates and thresholds were fixed before scoring. All four gates pass at k=18 against the same LeMiCa arm. On pairs where the dense model is dynamic, motion retention is 0.940 against a 0.90 floor, and the motion-lost fraction is 0.060 against a 0.10 ceiling. Dynamic degree is within 0.02 of LeMiCa on 173 of 176 pairs, exceeding the 80% floor. Subject consistency and temporal flickering are within 0.005 of LeMiCa on 94.3% and 97.7% of pairs, also above that floor. Imaging and aesthetic quality meet non-inferiority on 146 and 137 of 176 pairs, respectively, exceeding the 60% floor. PropCache leads LeMiCa by 1.5 raw VBench points on imaging quality and wins 157 of 198 non-tied pairs among the full 200, with $p < 10^{-16}$ in a two-sided sign test. The motion-collapse check passes. Table 8 reports Wan2.2 results by dimension.

Table 10: **Text-to-video, continued: HunyuanVideo-1.5 and the 1.3B Wan2.1.** Protocol, bolding and footnote marks follow Table 1. NFE counts forward passes; each budget group shares one wall-time measurement. The rerun MagCache results in Section J.9 lead both HunyuanVideo-1.5 groups. All methods in that block show reduced absolute fidelity, so the block supports relative comparisons within this evaluation regime.

Method	NFE	Latency (s)↓	Speedup↑	LPIPS↓	PSNR↑	SSIM↑
HunyuanVideo-1.5	16 prompts					
Dense (50 steps)	50	174.9	1.00×	–	–	–
TeaCache	20	69.8	2.50×	0.6195	12.3	0.354
MagCache	20	69.8	2.50×	**0.1878**	**21.3**	**0.759**
BudCache	20	69.8	2.50×	0.3428	16.1	0.583
LeMiCa	20	69.8	2.50×	0.2949	17.7	0.638
PropCache	20	69.8	2.50×	0.2092	20.2	0.724
TeaCache	25	87.3	2.00×	0.5638	12.4	0.379
MagCache	25	87.3	2.00×	**0.1499**	**22.3**	**0.794**
BudCache	25	87.3	2.00×	0.3394	16.0	0.575
LeMiCa	25	87.3	2.00×	0.1824	21.3	0.760
PropCache	25	87.3	2.00×	0.1555	22.0	0.790
Wan2.1-T2V-1.3B	6 held-out prompts					
Dense (50 steps)	50	100.1	1.00×	–	–	–
DBCache†	10.2	–	–	0.7197	11.9	0.381
TeaCache	14	27.7	3.61×	0.1997	19.2	0.716
BudCache	14	27.7	3.61×	0.2617	18.3	0.643
LeMiCa	14	27.7	3.61×	0.1536	21.5	0.790
PropCache	14	27.7	3.61×	**0.1414**	**23.3**	**0.820**
TeaCache†	22	–	–	0.1316	20.9	0.785
BudCache	25	49.5	2.02×	0.1989	18.8	0.709
LeMiCa	25	49.5	2.02×	0.0340	30.1	0.945
PropCache	25	49.5	2.02×	**0.0293**	**31.2**	**0.958**
DBCache†	26	–	–	0.2929	16.9	0.641

F THRESHOLD-TRIGGERED ARMS AT MEASURED COMPUTE

Threshold-triggered arms have compute costs determined by their trigger rules. This section compares them at their measured operating points on two backbones.

Wan2.1-T2V-14B. Table 2 reports five budget-exact tiers, each executing exactly k forwards. DBCache executes 15.9 forwards, between tiers. MagCache matches $k{=}20$ and $k{=}25$, as reported in Table 2; its setting calibrated for tighter budgets executes 19.0 forwards. Table 11 reports these arms at their realized compute. PropCache at $k{=}17$ has lower error and is 10% faster than MagCache at 19 forwards, with error lower than DBCache by a factor of 6. TeaCache's two tested thresholds execute exactly 17 and 20 forwards, permitting post-hoc budget-matched comparisons. PropCache reduces error by 41% and 54% at those compute levels.

Arm	Realized NFE	Latency (s)↓	Speedup↑	LPIPS↓
DBCache ($\tau{=}0.14$)	15.9 (15–20)	161.6	3.02×	0.4848
MagCache ($\tau{=}0.20$)	19	188.7	2.59×	0.0809
TeaCache ($\tau{=}0.20$)	17	169.1	2.89×	0.1337
TeaCache ($\tau{=}0.14$)	20	198.5	2.46×	0.1209
PropCache ($k{=}17$)	17	169.6	2.88×	0.0784
PropCache ($k{=}20$)	20	198.9	2.45×	**0.0555**

Table 11: **Wan2.1-T2V-14B: methods outside the budget-exact group.** Means use the same 16 held-out prompts, dense reference and evaluation harness as Table 1. NFE counts executed forward passes; latency is measured wall time. DBCache's NFE varies across prompts; its entry gives the mean and range.

Wan2.2-T2V-A14B block-level compute and latency. The campaign recorded wall clock and executed transformer blocks for six threshold arms; per-arm forward counts are unrecorded. Table 13 marks their NFE entries with dashes, and Figure 6 includes arms with measured NFE ratios. Exact conversion from block counts to NFE requires whole-step skipping; the two DBCache settings use intra-step skipping.

Measured block counts determine FLOPs for all twelve arms in Table 13. The six threshold arms use 60.4 to 91.3 P, compared with 53.7 P for PropCache at $k{=}16$. TeaCache at $\tau{=}0.08$ has the lowest threshold-arm cost, matching the 60.4 P of the budget-exact $k{=}18$ group, with LPIPS 0.5717 against 0.0752 for PropCache. Figure 18 compares all twelve arms by measured end-to-end wall clock using the same dense reference, 16 prompts and harness. Compute and latency rankings agree for the budget-exact arms in Table 1. The six threshold arms achieve $1.47\times$ to $2.21\times$ speedup with LPIPS 0.1182 to 0.5717; PropCache achieves 0.0952 at $2.46\times$. Under the marking rule of Section J.9, budget-exact classification requires forward-counted runs on the same prompts and seeds.

G PER-BUDGET SELECTION AND REPLAY-GRAPH MASK AGREEMENT

Section 4.2 compares a replay-graph scheduler with the $r{=}\infty$ member of our family. Both minimize the largest edge cost along a path. The replay-graph scheduler uses replayed edge measurements and lexicographic tie-breaking; PropCache uses the largest analytic step price. Identical weights and a unique bottleneck minimizer imply identical paths. With independently constructed weights, mask agreement is evaluated empirically.

Table 12 reports mask agreement and quality on FLUX, HunyuanVideo, abbreviated HYV, and HunyuanVideo-1.5. It covers ten budgets passing a predefined intake gate of 2% mean gain over the canonical cost on held-out verification pools. Quality is measured by LPIPS-alex against a same-process dense reference at equal budget $n_{\mathrm{calc}}{=}k$; † marks an identical baseline mask. HYV-1.5 uses a six-unit one-step probe set for per-unit pricing and 24 units for pooled pricing. equation 7 uses the calibration pool for both.

Among the admitted tiers, the replay-graph baseline has 4.4% lower mean error at one video budget. At one FLUX budget, a search-based baseline has lower mean error while PropCache wins 13 of

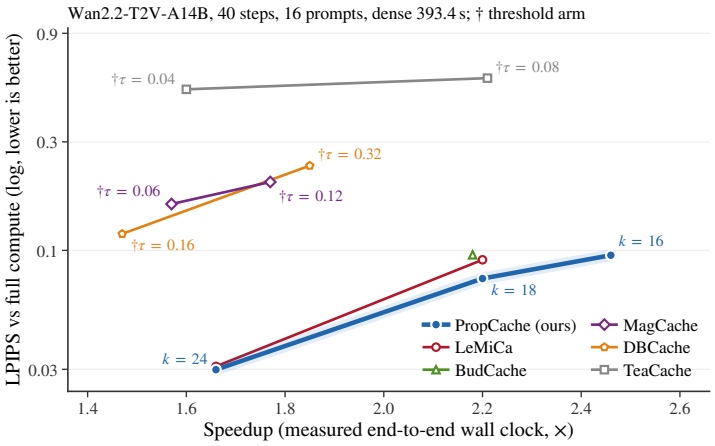

Figure 18: **Wan2.2-T2V-A14B on measured wall clock.** End-to-end speedup for all twelve arms of Table 13. The six threshold arms have recorded latency and block counts; their forward counts are unrecorded. Daggers mark their trigger thresholds τ. LeMiCa is evaluated at k=18 and 24, and BudCache at k=18.

model	k	dim*	r^*	vs. canonical	vs. replay-graph	paired wins
FLUX (n=28, 24 units)	12	pool	2.2	+18.8%	+0.9%	21/24, 17/24
FLUX	14	unit	4.0	+19.0%	0.0%†	22/24, —
FLUX	18	unit	2.2	+23.9%	0.0%†	21/24, —
FLUX	22	unit	2.0	+23.8%	0.0%†	17/24, —
HYV (n=50, 10 units)	16	pool	2.4	+6.9%	+2.9%	7/10, 7/10
HYV	18	pool	3.6	+5.6%	+5.6%	5/10, 8/10
HYV	25	unit	5.0	+11.1%	−4.4%	6/10, 6/10
HYV-1.5 (n=50, 16 units)	14	unit	2.0	+7.6%	+25.1%	12/16, 15/16
HYV-1.5	20	unit	2.6	+3.7%	+29.1%	12/16, 16/16
HYV-1.5	25	unit	2.0	+5.6%	+14.8%	9/16, 14/16

Table 12: Per-budget selection on three models. Gains are verification-pool means relative to the canonical cost at r=2 with pooled pricing and to the replay-graph scheduler at the same budget.

24 paired units. Three FLUX masks match the replay-graph masks exactly under the predefined compile rule.

H Extended Video and Image Results

This section reports the full text-to-video campaign and three additional audits referenced in Section 5.

Wan2.1-T2V-14B at k=17. Section 5.1 reports the whole-suite comparison with LeMiCa at k=14. On the same 943 pairs at k=17, LeMiCa's mean LPIPS is 6×10^{-4} lower, while PropCache has a 44.1% paired win rate. The mean gap is one eighth of the tie band, with p=0.059. The k=14 advantage for PropCache has $p = 2 \times 10^{-6}$ under the same test.

Objective transfer to a disjoint cohort. The bit-identical recovery in Section 5.1 uses Wan2.2's historical cohort. A separate split uses 24 calibration trajectories and 16 disjoint evaluation units. At k=18, the deployed minimax objective beats the additive alternative on 15 of 16 units, including all 8 of 8 units with unseen prompts. The paired mean LPIPS reduction is 1.63×10^{-3}, with a 95% bootstrap CI for the signed difference of $[-2.5, -0.7] \times 10^{-3}$. At k=24, both objectives are within 0.03 LPIPS of dense and tie.

FLUX threshold-arm comparisons. At 14 forwards, PropCache runs 11% faster than TeaCache with 61% less error. Compared with MagCache at 9 forwards, PropCache uses one additional forward and 5% more wall time to reduce error by 30%.

Table 13: **Complete text-to-video results on 14B-class backbones.** Compute is matched within each group; the main text presents the binding-budget groups and flagship ladder. **Bold** marks the per-group best. NFE counts executed forward passes. FLOPs count transformer matrix multiplications across both classifier-free guidance branches. For the six Wan2.2 threshold methods, FLOPs use measured block-equivalent counts from executed-block logs; see Section J.9. Latency is denoising-loop wall time; ^{e} denotes end-to-end timing. † marks a triggered method outside the matched group at its realized compute. DBCache is off-budget at every Wan2.1-T2V-14B rung; see Section F.

		Efficiency			Visual quality		
Method	NFE	FLOPs (P)↓	Latency (s)↓	Speedup↑	LPIPS↓	PSNR↑	SSIM↑
Wan2.2-T2V-A14B^{e} 16 prompts							
Dense (40 steps)	40	134.3	393.4	1.00×	–	–	–
DBCache (τ=0.16)	–	91.3	267.9	1.47×	0.1182	23.3	0.843
MagCache (τ=0.06)	–	85.6	251.4	1.57×	0.1599	21.7	0.781
TeaCache (τ=0.04)	–	83.9	246.3	1.60×	0.5105	13.3	0.461
MagCache (τ=0.12)	–	75.5	222.0	1.77×	0.1998	21.7	0.745
DBCache (τ=0.32)	–	72.5	213.0	1.85×	0.2357	19.6	0.694
TeaCache (τ=0.08)	–	60.4	178.2	2.21×	0.5717	13.1	0.442
PropCache	16	53.7	159.9	2.46×	**0.0952**	**26.6**	**0.866**
BudCache	18	60.4	180.2	2.18×	0.0956	27.2	0.865
LeMiCa	18	60.4	178.5	2.20×	0.0909	27.3	0.873
PropCache	18	60.4	178.5	2.20×	**0.0752**	**28.2**	**0.906**
LeMiCa	24	80.6	237.0	1.66×	0.0309	32.7	0.952
PropCache	24	80.6	237.1	1.66×	**0.0299**	**32.9**	**0.961**
Wan2.2-T2V-A14B, disjoint split^{e} 16 unseen units, 24-unit calibration							
Dense (40 steps)	40	134.3	392.3	1.00×	–	–	–
BudCache	18	60.4	176.8	2.22×	0.0967	27.6	0.865
LeMiCa	18	60.4	176.9	2.22×	0.0932	27.6	0.867
PropCache	18	60.4	176.9	2.22×	**0.0786**	**28.2**	**0.892**
BudCache	24	80.6	235.1	1.67×	0.0402	32.4	0.945
LeMiCa	24	80.6	235.4	1.67×	0.0342	33.1	0.951
PropCache	24	80.6	235.5	1.67×	**0.0317**	**33.2**	**0.955**
Wan2.1-T2V-14B^{e} 16 prompts; full 943-prompt suite in text							
Dense (50 steps)	50	167.8	488.3	1.00×	–	–	–
PropCache	10	33.6	101.2	4.83×	0.1801	21.4	0.746
TeaCache	14	47.0	139.0	3.51×	0.1657	21.2	0.757
BudCache	14	47.0	139.0	3.51×	0.1174	23.4	0.814
LeMiCa	14	47.0	139.0	3.51×	0.1260	22.9	0.804
PropCache	14	47.0	139.0	3.51×	**0.1124**	**24.0**	**0.829**
TeaCache	17	57.1	168.1	2.90×	0.1337	22.2	0.790
BudCache	17	57.1	168.1	2.90×	**0.0783**	**26.3**	**0.885**
LeMiCa	17	57.1	168.1	2.90×	0.0808	25.2	0.871
PropCache	17	57.1	168.1	2.90×	0.0784	25.9	0.881
TeaCache	20	67.1	197.2	2.48×	0.1209	22.5	0.798
BudCache	20	67.1	197.2	2.48×	0.1612	20.6	0.749
MagCache	20	67.1	197.2	2.48×	0.0741	25.2	0.875
LeMiCa	20	67.1	197.2	2.48×	**0.0552**	27.3	0.912
PropCache	20	67.1	197.2	2.48×	0.0555	**27.5**	**0.914**
TeaCache	25	83.9	245.5	1.99×	0.0662	25.6	0.887
BudCache	25	83.9	245.5	1.99×	0.2120	18.9	0.685
MagCache	25	83.9	245.5	1.99×	0.0549	26.5	0.905
LeMiCa	25	83.9	245.5	1.99×	**0.0317**	29.8	0.945
PropCache	25	83.9	245.5	1.99×	**0.0317**	**29.9**	**0.946**
HunyuanVideo-13B 16 prompts							
Dense (50 steps)	50	62.3	93.5	1.00×	–	–	–
BudCache	16	19.9	29.8	3.14×	0.1530	25.2	0.860
LeMiCa	16	19.9	29.9	3.12×	0.1027	29.9	0.923
PropCache	16	19.9	29.9	3.13×	**0.0803**	**30.3**	**0.928**
TeaCache†	16.2	20.2	30.9	3.03×	0.1996	22.8	0.802
MagCache†	18	22.4	33.7	2.77×	0.0976	28.5	0.889
TeaCache†	24.9	31.0	46.9	1.99×	0.1621	23.8	0.840
MagCache	25	31.1	46.7	2.00×	0.0484	32.4	0.947
BudCache	25	31.1	46.6	2.01×	0.1285	26.5	0.870
LeMiCa	25	31.1	46.8	2.00×	0.0244	36.7	0.977
PropCache	25	31.1	46.7	2.00×	**0.0211**	**37.0**	**0.979**

Table 14: **HunyuanVideo-13B: speedup and fidelity by budget.** Means over 16 held-out units; LPIPS, SSIM and PSNR use the dense reference. Wall speedup uses end-to-end latency; NFE speedup uses forward-pass counts. The final columns compare MagCache at PropCache's wall time, which is unreachable for MagCache at $k \leq 10$. Δ is the mean per-unit relative LPIPS change.

Budget	Wall spd.	NFE spd.	LPIPS↓	SSIM↑	PSNR↑	vs. MagCache at matched wall time	
k	(×)	(×)			(dB)	LPIPS↓	Δ
8	4.40	6.25	0.282	0.814	24.55	–	unreachable
10	3.79	5.00	0.209	0.850	25.73	–	unreachable
12	3.32	4.17	0.149	0.885	27.56	0.333	-55.5%
14	2.97	3.57	0.103	0.910	29.07	0.184	-43.1%
16	2.67	3.12	0.080	0.928	30.30	0.136	-35.4%
18	2.44	2.78	0.060	0.942	31.66	0.101	-33.8%
20	2.23	2.50	0.044	0.959	33.44	0.076	-33.6%
22	2.07	2.27	0.032	0.969	34.87	0.058	-33.7%
25	1.85	2.00	0.021	0.979	36.98	0.048	-43.0%

Table 15: **HunyuanVideo-13B: per-unit comparison with MagCache.** LPIPS-alex against dense for 16 held-out units at 9 budgets. MagCache is compared at PropCache's wall time using a ± 0.005 LPIPS tie band: plain entries indicate PropCache wins, underline ties, and **bold*** losses. †: outside MagCache's measured latency range.

Unit	k=8	k=10	k=12	k=14	k=16	k=18	k=20	k=22	k=25
01	0.299†	0.236†	0.131	0.086	0.057	0.038	0.027	0.020	0.013
02	0.499†	0.332†	0.202	0.123	0.087	0.048	0.037	0.028	0.016
03	0.290†	0.201†	0.127	0.091	0.074	0.065	0.053	0.044	0.030
04	0.199†	0.129†	0.101	0.083	**0.111***	0.091	0.053	0.057	0.044
05	0.243†	0.177†	0.134	0.094	0.070	0.051	0.052	0.024	0.021
06	0.269†	0.176†	0.131	0.093	0.056	0.040	0.024	0.016	0.011
07	0.366†	0.297†	0.224	0.177	0.137	0.115	0.084	0.060	0.048
08	0.143†	0.111†	0.071	0.048	0.036	0.025	0.018	0.015	0.008
09	0.322†	0.256†	0.210	0.157	0.105	0.059	0.048	0.028	0.028
10	0.301†	0.234†	0.186	0.082	0.078	0.065	0.048	0.029	0.012
11	0.245†	0.189†	0.105	0.071	0.058	0.042	0.027	0.021	0.017
12	0.272†	0.234†	0.150	0.098	0.067	0.043	0.039	0.024	0.013
13	0.469†	0.313†	0.232	0.159	0.132	0.109	**0.085***	**0.073***	0.030
14	0.079†	0.063†	0.044	0.038	0.022	0.016	0.013	0.011	0.005
15	0.239†	0.181†	0.148	0.121	0.094	0.080	0.044	0.031	0.022
16	0.283†	0.217†	0.182	0.126	0.100	0.081	0.054	0.036	0.020

I Extended Tables and Provenance

I.1 Provenance index

Tables 16 and 17 link 31 key values in the abstract, Section 5 and Section 6 to their archived source records. `build_provenance.py` recomputes each entry and fails the build on a mismatch. Eleven rows aggregate per-unit arm records, including the nine-budget HunyuanVideo-13B comparison, the disjoint-split pairing and the FLUX per-pair median. The remaining rows are checked against values produced by the table and figure generators.

J Propagation Price: Derivations, Protocols and Audits

J.1 Gain Profile: Protocol and Per-Model Audits

The one-step probe caches only step t, computes all other steps, and records the terminal displacement. Dividing its norm by the injected defect norm gives $A(t)$. A geometric factor in the step index

Table 16: **Provenance index, part 1 of 2: flagship ladder, full prompt suite and disjoint split.** Records and generators for numerical claims in the abstract, §5 and §6. `P/` denotes the paper package; `D/` denotes the released measurement archive. `build_provenance.py` recomputes all 31 rows across both parts and halts the build on any mismatch.

Claim	As printed	Where	Record	Generator
One calibration recipe covers six flow-matching DiTs	6 models	Abs., Tab. 1–3	`P/figdata/main_table_{video,image}.json`	`build_main_tables.py`
Budget-exact groups where PropCache leads LPIPS, PSNR and SSIM outright	10 of 14	§5, Tab. 1	`P/figdata/main_table_video.json (audit)`	`build_main_tables.py`
The two remaining LPIPS cells, both inside the ± 0.005 tie band	1×10^{-4}, 3×10^{-4}	§5	`P/figdata/main_table_video.json (audit)`	`build_main_tables.py`
PropCache's worst gap to the per-budget best over the flagship ladder	3×10^{-4}	§5, §6	`P/figdata/main_table_video.json (ladder)`	`build_main_tables.py`
LeMiCa's worst gap over the same four rungs, at k=14	0.0136	§5, §6	`P/figdata/main_table_video.json (ladder)`	`build_main_tables.py`
BudCache's worst gap at its loosest budget, k=25	0.1803	§5, §6	`P/figdata/main_table_video.json (ladder)`	`build_main_tables.py`
TeaCache's worst gap over the same rungs	0.0657	§5	`P/figdata/main_table_video.json (ladder)`	`build_main_tables.py`
Ladder rungs reachable by MagCache's threshold	2 of 4	§5	`P/figdata/main_table_video.json (ladder)`	`build_main_tables.py`
Fastest flagship rung; no baseline in the campaign reaches it	k=10, 4.83×	Tab. 1	`P/figdata/main_table_video.json`	`build_main_tables.py`
Whole-suite mean LPIPS at k=14, PropCache vs. LeMiCa	0.0966 / 0.0999	§5	`P/figdata/wan21_943_suite.json`	`wan21_943_suite.py`
Centred paired bootstrap on all 943 prompts at k=14	p=2×10^{-6}	Abs., §5	`P/figdata/wan21_943_suite.json`	`wan21_943_suite.py`
Same test at k=17: LeMiCa leads with a nonsignificant difference	p=0.059	§5	`P/figdata/wan21_943_suite.json`	`wan21_943_suite.py`
Iso-wall check on the 943-prompt pair	140.49 / 140.44 s	§5	`P/figdata/wan21_943_suite.json`	`wan21_943_suite.py`

times an exponential in σ_t fits the decay with R^2 from 0.911 to 0.928 across statistically indistinguishable parameter pairs. Fitted exponents vary with the noise schedule and probe amplitude. The solver uses the measured profile directly, relying on its overall decay and rank validity. The profile-shape ablation in Section 5.3 tests sensitivity to distortions; Table 18 reports per-model decay and rank fits.

The scalar $A(t)$ pools gains along the probed defect directions at step t. Per-prompt profile correlations and independent price-rank fits assess this reduction. The matrix propagator below retains direction dependence.

J.2 From One Defect to Many: The Mask-Dependent Propagator

Extending the one-step price in equation 3 to a multi-step schedule requires the two corrections stated in Section 3.4. Linear theory accounts for mask-dependent propagation and contribution geometry beyond summing prices or adding them in quadrature; Section 4.2 calibrates the resulting cost. Let δ_t denote the cached trajectory's deviation from the reference. Under the same linearization, a *computed* step advances it by $\delta_{t+1} = (I + \Delta\sigma_t J_t)\delta_t$, while a *cached* step uses the anchor's velocity:

$$\delta_{t+1} = \underbrace{I\,\delta_t}_{\text{one-step factor}\to I} + \Delta\sigma_t\Big[\underbrace{\xi_t}_{\text{injection}} + \underbrace{J_{a(t)}\,\delta_{a(t)}}_{\text{anchor drift}}\Big], \qquad m_t = 0. \tag{9}$$

A cached step injects the local defect ξ_t of equation 2 and transports the accumulated deviation through the identity plus first-order anchor drift. A computed step applies the factor $I + \Delta\sigma_t J_t$. Iterating equation 9 over a mask gives

$$e_n = \sum_{t:\, m_t=0} \Delta\sigma_t\, \Phi_m(n, t{+}1)\, \xi_t + O\Big(\textstyle\sum_t \|\xi_t\|^2\Big), \tag{10}$$

Table 17: **Provenance index, part 2 of 2: cross-model, cross-modality and audit claims.** Continuation of Table 16, with the same columns and build-time verification.

Claim	As printed	Where	Record	Generator
Per-category gain and win rate, *human action*	$+18.0\times10^{-3}$, 82%	Fig. 17	`P/figdata/wan21_943prompt_final_verdict.json`	`fig_wan21_category_win.py`
Wan2.2: compiled mask vs. the mask exhaustive per-step measurement selects	identical, 16/16	Abs., §5	`D/wan2.2-14B/.../table_minimax.json`	--
Rank agreement between the closed-form price and the measured masks	Spearman 0.857	§5	`D/wan2.2-14B/.../table_minimax.json`	--
Wan2.2 disjoint split, 24 calibration vs. 16 unseen evaluation units: minimax vs. additive	-1.63×10^{-3}, 15/16	§5	`D/wan2.2-14B/canonical_scale/verdict.json`	--
Wan2.2 disjoint split at k=18: PropCache / LeMiCa / BudCache	.0786 / .0932 / .0967	Abs., §5, Tab. 1	`P/figdata/wan22_disjoint_h2h.json`	`wan22_disjoint_h2h.py`
… per-unit wins and paired bootstrap against each	15/16, 14/16	§5	`P/figdata/wan22_disjoint_h2h.json`	`wan22_disjoint_h2h.py`
… at k=24, and both groups swept on all three metrics	.0317 / .0342 / .0402	§5, Tab. 1	`P/figdata/wan22_disjoint_h2h.json`	`wan22_disjoint_h2h.py`
Wan2.2 VBench dimensions PropCache scores above LeMiCa on	5 of 6	Tab. 8	`P/figdata/vbench_wan22.json`	`build_vbench_table.py`
HunyuanVideo-13B fastest audited rung by wall time, 16 units	4.40×	Abs., Tab. 14	`P/hyv_v2_audit/verdict.json`	`make_appendix_tables.py`
HunyuanVideo-13B vs. LeMiCa at nine budgets, LPIPS reduction	13.5–31.8%	Abs., §5	`D/hyv-13B/hyv_baselines_v2_arms/`	`build_provenance.py`
… mean over the nine budgets, per-unit wins, wall agreement	23%; 9–16/16; 0.3%	§5	`D/hyv-13B/hyv_baselines_v2_arms/`	`build_provenance.py`
HunyuanVideo-13B vs. the per-budget search at k=16	47%	Abs., Tab. 1	`P/figdata/main_table_video.json`	`build_main_tables.py`
HunyuanVideo-13B iso-latency audit: PropCache wins / ties / losses	350 / 12 / 4 of 432	Tab. 9	`D/hyv-13B/hyv13_baselines_v2_verdict_*.json`	`build_audit_totals.py`
… unreachable baseline operating points, excluded from win/tie/loss counts	66	Tab. 9	`D/hyv-13B/hyv13_baselines_v2_verdict_*.json`	`build_audit_totals.py`
FLUX budget-exact group at k=14, PropCache vs. the best baseline	0.0827 / 0.0878	Tab. 3	`P/figdata/main_table_image.json`	`build_main_tables.py`
FLUX vs. the trigger baseline, per-pair median LPIPS reduction	−62% (n=176)	Abs., §5	`D/flux-12B/scores/fidelity_scores.json`	`build_provenance.py`
Models the dimensionless coherence constant places in one regime band	6 of 6	Abs., Fig. 12	`P/figdata/regime_table.json + 2 more`	`fig_regime_six.py`
Wan2.2 transformer FLOPs: the τ=0.08 trigger and the budget-exact k=18 group	60.4 / 60.4 P	Tabs. 1, 13	`P/figdata/flops_model.json`	`flops_model.py`

Table 18: **Offline calibration measurements by backbone.** *Start* and *end* are gain-profile endpoints on a 12-point grid; *decay* is the max/min ratio. The right block fits predicted price against measured single-step error in log–log space on a held-out sweep. All four fits satisfy the prespecified gates. "Inst." denotes I1 velocity-space or I2 latent-space. †HunyuanVideo-1.5 stores a relative endpoint defect; cross-row comparisons use its profile shape.

				Gain profile $A(t)$			Price rank fit		
Backbone	Inst.	Steps	Units	start	end	decay	slope	R^2	n
Wan2.1-T2V-14B	I1	50	6	42.2	0.84	×54	0.76	0.77	784
Wan2.1-T2V-1.3B	I1	50	6	87.0	0.91	×96	0.72	0.82	72
Wan2.2-T2V-A14B	I2	40	8	47.7	0.75	×64	0.97	0.83	18
HunyuanVideo-13B	I2	50	2	95.2	0.89	×107	0.87	0.86	24
HunyuanVideo-1.5†	I2	50	6	0.5	0.05	×11	–	–	–
FLUX.1-dev	I1	28	12	54.3	2.27	×24	–	–	–

where the *mask-dependent* propagator Φ_m solves equation 9 on the augmented state (δ_t, c_t), with c_t storing the cached anchor's deviation. Computed steps apply the one-step factor; cached steps apply the identity to the running deviation and transport $J_{a(t)}\delta_{a(t)}$ through the anchor channel. This channel closes the recursion by carrying earlier defects through the anchor's velocity at later cached steps. At first order and a fixed mask, defects *superpose* as a vector sum of terminal contributions. With $h_j = \Delta\sigma_j$, $F_j = I + h_j J_j$, and $z_j = (\delta_j, c_j)$, where c_j stores the anchor's deviation on entry

to step j, the homogeneous transitions are

$$M_j^{\text{compute}} = \begin{pmatrix} F_j & 0 \\ I & 0 \end{pmatrix}, \qquad M_j^{\text{cache}} = \begin{pmatrix} I & h_j J_{a(j)} \\ 0 & I \end{pmatrix}.$$

A cached-step injection enters through $B = (I, 0)^\top$, and the terminal latent is read by $P = (I, 0)$, giving $\Phi_m(n, t+1) = PM_{n-1} \cdots M_{t+1}B$. The ordered product retains both the running deviation and the anchor drift.

Comparing the two propagators. A one-step probe sees the dense propagator of equation 3; a mask realizes Φ_m. The two differ only at the steps the mask caches, and there the difference telescopes exactly: with $\Pi_j = F_{n-1} \cdots F_{j+1}$ the computed tail and $Z_j = M_{j-1} \cdots M_{t+1}B$ the augmented state the mask delivers to step j,

$$\Phi(n, t{+}1) - \Phi_m(n, t{+}1) = \sum_{j > t:\, m_j = 0} \Pi_j \, h_j \big(J_j \;\; -J_{a(j)}\big) Z_j, \tag{11}$$

where the block row $(J_j \;\; -J_{a(j)})$ acts on the running deviation and stored anchor, respectively. Each cached step contributes one term. The anchor row of $M_j^{\text{compute}} - M_j^{\text{cache}}$ drops out because the computed tail reads only the deviation slot. For an injected direction r, the local term is $h_j(J_j\delta_j - J_{a(j)}c_j)$, with $(\delta_j, c_j) = Z_j r$. Its decomposition, $h_j[(J_j - J_{a(j)})\delta_j + J_{a(j)}(\delta_j - c_j)]$, captures both Jacobian variation and deviation accumulated since the anchor, including when the Jacobians agree. Equation 11 sums these terms through the computed tail. Each carries h_j and is first order in the local step size. The executed-mask measurements in Section J.3 include changes to both propagation and injected defects.

Separating price scale and geometry. With $y_i = \Delta\sigma_i \Phi_m(n, i+1)\xi_i$ the terminal contributions of a fixed linearized mask and ℓ_i their prices, factorization gives

$$\frac{\sum_i \ell_i}{\|\sum_i y_i\|} = \underbrace{\frac{\sum_i \ell_i}{\sum_i \|y_i\|}}_{\text{price-to-contribution ratio}} \cdot \underbrace{\frac{\sum_i \|y_i\|}{\|\sum_i y_i\|}}_{\text{triangle slack}}, \tag{12}$$

for nonzero denominators. The first factor compares scalar prices with the norms of the fixed-mask contributions; the second depends on their directions. The interference experiment below gives a separate, directly observable factorization using the individual and joint terminal responses.

J.3 Simultaneous Injection: The Interference Measurement

For each of 12 probed direction pairs on Wan2.1-T2V-1.3B at natural defect strength, we compare the joint terminal error norm with the sum of the individual terminal error norms. The interaction coefficient is negative in all 12 pairs, and 83–97% of samples reverse the sign predicted by the additive model, indicating saturation under simultaneous injection.

Let δ_s and δ_t be the measured terminal error vectors from the two individual cache interventions, and $\delta_{s,t}$ the terminal error from their joint intervention. Their measured norms obey the identity

$$\frac{\|\delta_s\| + \|\delta_t\|}{\|\delta_{s,t}\|} = \underbrace{\frac{\|\delta_s + \delta_t\|}{\|\delta_{s,t}\|}}_{\text{response ratio}} \underbrace{\frac{\|\delta_s\| + \|\delta_t\|}{\|\delta_s + \delta_t\|}}_{\text{triangle slack}}. \tag{13}$$

The response ratio compares three executed masks and captures changes in propagation and injected defects under joint intervention. The triangle slack uses the two individually measured vectors. Equation 12 factorizes contributions within a fixed mask; this experiment evaluates equation 13 across masks.

step pair	composite	response ratio	slack	ρ measured	ρ needed
$(1, 13)$	1.03–1.09	1.00–1.01	1.03–1.08	$+0.02$–0.06	-0.02–-0.03
$(13, 19)$	1.93–1.98	1.65–1.68	1.17–1.18	$+0.43$–0.47	-0.46–-0.50
$(19, 25)$	1.87–1.94	1.60–1.69	1.15–1.17	$+0.45$–0.51	-0.43–-0.48
$(25, 31)$	1.87–1.97	1.59–1.70	1.16–1.18	$+0.44$–0.49	-0.43–-0.48
$(31, 38)$	1.95–1.97	1.67–1.70	1.16–1.16	$+0.48$–0.48	-0.47–-0.49
$(13, 38)$	1.82–1.86	1.53–1.58	1.18–1.19	$+0.40$–0.44	-0.41–-0.43
median over all 12	1.901	1.625	1.165	$+0.448$	-0.448

Table 19: **Aggregation exponents: reference cases and calibrated selections.** The exponent in equation 5 controls within-segment aggregation. The table summarizes reference cases for vector-sum norms and selections from the evaluated grid. Per-backbone probe statistics and derivations appear in Appendices J.4, J.6 and J.3.

Quantity	Value	Interpretation
Reference exponents		
$r = 1$, sum	aligned vectors	the sum of contribution norms upper-bounds linearized error
$r = 2$, quadrature	orthogonal vectors	quadrature lower-bounds linearized error under nonnegative pairwise overlaps
$r = \infty$, largest step	one nonzero vector	each analytical edge uses its largest step price, matching the replay-graph objective level
Calibrated grid selections		
Selected exponent, mean	$\bar{r} = 2.8$	$7/10$ tiers above r=2; the composition fit gives $\hat{r} = 3.0$
Masks equal to replay baseline	$3/10$ tiers	analytic pricing reproduces the replayed compute mask bit for bit
Distinct members selected	8 across 10 tiers	no family member is selected more than 3 times
Calibrating both axes per budget	19% at k=14	either axis alone recovers 0.2%; see Section J.7
Compile / serve cost	$O(n^2k)$ / one lookup	n steps, budget k; no per-request search

Each row covers two units at natural defect strength. The composite is $(\|\delta_s\| + \|\delta_t\|)/\|\delta_{s,t}\|$, equal to response ratio $\times$ slack per pair. The column "ρ needed" gives the cosine required to reproduce the joint norm using the individual response norms: $(\|\delta_{s,t}\|^2 - \|\delta_s\|^2 - \|\delta_t\|^2)/(2\|\delta_s\|\|\delta_t\|)$. The geometric slack is at most 1.20 and accounts for $\log 1.165/\log 1.901 = 24\%$ of the log-ratio using the reported medians.

On equal-magnitude pairs, the slack approaches the geometric bound $2/\sqrt{2+2\rho}$. Evaluated at each mid-band pair's cosine, this bound is 1.15–1.20, compared with measured slack of 1.15–1.19 on those five pairs. For $(1, 13)$, the nearly orthogonal directions at $\rho \approx 0.02$–0.06 give an equal-magnitude bound of 1.37–1.40. Its measured slack of 1.03–1.08 reflects unequal contribution magnitudes: for $\|\delta_s\| \gg \|\delta_t\|$, the general slack $(\|\delta_s\| + \|\delta_t\|)/\|\delta_s + \delta_t\|$ tends to 1 at any cosine. The bound is attained under its equal-magnitude condition. The required and measured cosines have opposite signs, with medians -0.448 and $+0.448$ and separations of 0.81–0.99 on the ten mid-band pairs, identifying a response change beyond the measured geometry.

The response ratio is 1.5–1.7 on the ten mid-band pairs. Adjacent profile ratios $A(t-1)/A(t)$ are smaller: $1.138, 1.086, 1.081, 1.040, 1.015$ at $t = 13, 19, 25, 31, 38$. The largest occurs at $t = 13$, while the early pair $(1, 13)$ has the smallest response ratio, 1.00–1.01. Equation 11 captures the running and anchor states through $\Pi_j h_j (J_j - J_{a(j)}) Z_j$. Joint interventions change these transitions and can change subsequent defect anchors. Equation 13 measures the resulting response ratio; Section 4.2 selects the deployed exponent by calibration LPIPS. These measurements support the beyond-quadrature branch of equation 5 with positive measured cosines throughout.

J.4 Admissible Exponent Range and Directional Overlap

The contribution envelope supports Section 4.1 and defines the overlap statistic in Table 19.

At a fixed mask, let $E_{\text{lin}}(a, t)$ be the norm of the linearized terminal contribution from the segment's injected defects. Let ν collect the per-step contribution norms $\|\Delta\sigma_t \Phi_m(n, t+1)\xi_t\|$, denoted $\|y_i\|$ in equation 12. The solver's price vector q is related to these norms through the price-to-contribution ratio. For the Gram matrix G of their unit directions, G_{ii}=1, and

$$E_{\text{lin}}(a,t)^2 = \nu^\top G\,\nu, \qquad \nu \geq 0, \quad G_{ij} \geq 0 \quad \Longrightarrow \quad \|\nu\|_2 \;\leq\; E_{\text{lin}} \;\leq\; \|\nu\|_1. \tag{14}$$

Expanding $\nu^\top G\nu$ gives diagonal sum $\|\nu\|_2^2$ and nonnegative off-diagonal terms under $\nu \geq 0$, $G_{ij} \geq 0$, proving the lower bound. The triangle inequality gives the upper bound. The condition $G_{ij} \geq 0$ applies pairwise to active contribution directions; norms are nonnegative by construction. Continuity and monotonicity of $r \mapsto \|\nu\|_r$ ensure that some $r^\star(a, t) \in [1, 2]$ reproduces E_{lin} exactly, uniquely whenever two contributions are nonzero. We call $[1, 2]$ the **admissible range**. The calibrated cost in Section 4.1 uses exponents above this range. The range's left endpoint alone provides a uniform upper bound under the stated hypotheses; on the same ν, every $r > 2$ gives a norm at most $\|\nu\|_2$.

The solver applies the exponent to the price vector q. Its scale relative to the fixed-mask contribution norms is the first factor of equation 12. For any nonzero nonnegative price vector of length L and $r \geq 1$, norm comparison gives

$$1 \;\leq\; \frac{\|q\|_1}{\|q\|_r} \;\leq\; L^{\,1-1/r}. \tag{15}$$

This bounds the change in price-based segment cost as r varies. The method selects one r per budget by evaluating masks with equation 7. Seven of the ten selected exponents exceed 2. The 90 paired interventions yield a median sum-to-joint-error ratio of 1.47, and the two-window fit gives $\hat{r} = 3.0$; see Section J.5.

From each one-step probe's terminal error direction $\hat{d}_t$, define the **attractor key**

$$\mathcal{A} = \frac{2}{L(L-1)} \sum_{t<t'} \hat{d}_t^{\top} \hat{d}_{t'} \tag{16}$$

as the unweighted mean pairwise cosine on a specified probe grid. Across all six image and video DiTs, from 1.3B to 14B, we measure $\mathcal{A} \in [0.10, 0.234]$. The contribution energy in equation 14 weights cosines by $\nu_i \nu_j$; for equal contribution norms, the mean cosine determines the energy directly, as derived in Section J.6. The sentinel $\mathcal{A} \geq 0.5$ triggers verification before routing a new model.

J.5 Two-Window Composition: Fit and Model Comparison

Composing separately measured high-noise and low-noise window costs with $C_{\text{joint}} = (C_h^r + C_l^r)^{1/r}$ gives $\hat{r} = 3.0$, a near-flat interval of $[2.9, 3.1]$, an in-sample residual $\leq 4.3\%$, and a held-out residual $\leq 2.8\%$. The fit uses Wan2.2-T2V-A14B's 40-step Euler flow-matching protocol, one source and one seed, and 11 joint configurations of two disjoint cached windows. The wider 90-pair set spans two video models. A constant Gram coefficient fits the same data at $\rho^*_{\text{eff}} = -0.19$, with pairwise values in $[-0.206, -0.147]$. Its AIC is comparable to the constant-exponent form, with $\Delta\text{AIC} \approx 1.8$. The identity $\rho_{\text{eff}}(a, b) = \left[(a^3 + b^3)^{2/3} - a^2 - b^2\right]/(2ab)$ defines this effective coefficient as the exponent-induced compression of the aggregation law. The individual terminal-response cosines are measured directly in Section J.3. The two-window estimate $\hat{r} = 3.0$ supports beyond-quadrature composition independently of the within-segment exponent calibrated by equation 7.

Geometric slack and response change. For two nonzero vectors with norms a, b and cosine $\rho > -1$, the triangle slack is

$$\frac{a+b}{\sqrt{a^2 + b^2 + 2\rho ab}} \leq \sqrt{\frac{2}{1+\rho}},$$

with equality at $a = b$, and at any a, b when $\rho = 1$. The bound follows by squaring and using $(1-\rho)(a-b)^2 \geq 0$. The left columns below evaluate this formula at reference cosine values drawn from the grid summaries. The right columns solve $\rho = 2/R^2 - 1$ for an equal-norm pair with slack R.

reference ρ	pair max slack	ratio to explain	equal-norm cosine
0.10, lowest full grid	1.348	1.35	$+0.097$
0.234, highest full grid	1.273	1.50	-0.111
0.26–0.47, mid band	1.260–1.166	$1.587 = 2^{2/3}$, fitted ℓ_3	-0.206
0.448, interference pairs	1.175	1.901, measured median	-0.447

Grid means summarize cosines across probes; each pair's slack uses its own two vectors. On the 12 interference pairs, slack is at most 1.20, with median 1.165, and the composite ratio has median 1.901. Equation 13 attributes the larger factor to terminal response change.

Across the 90-pair set, the composite ratio has median 1.47 and maximum 1.98; 55 ratios exceed the reference value 1.348. The joint terminal error is a median 1.008 times the larger individual error and is smaller in 28 of 90 pairs. For individual error norms a, b, $a^2 + b^2 + 2\rho ab \geq \max(a, b)^2$ whenever $\rho \geq 0$, so a two-vector Gram fit to those 28 norm triples requires a negative effective cosine. The 12 vector-resolved probes have positive measured cosines with median $+0.448$ and

required effective cosines with median -0.448. The deployed exponent captures the aggregation coefficient, while the measured cosines describe individual-response geometry.

For L equal-norm contributions with mean pairwise cosine $\mathcal{A}$, slack is $\sqrt{L/[1+\mathcal{A}(L-1)]}$. It equals 2.53 at $L = 16$, $\mathcal{A} = 0.10$, corresponding to $r_{\mathcal{A}} = 1.50$ in Section J.6. Multi-contribution geometry can therefore exceed the pairwise factors in the table. The exponent selected by equation 7 adjusts the segment cost within the band in equation 15.

J.6 Probe Protocol for the Coherence Key

ρ_{mid} and the attractor key $\mathcal{A}$ of equation 16 measure mean pairwise cosines of one-step terminal error directions on specified grids. On the smallest model, the mean is 0.14 on an early-step grid, 0.36–0.47 on the mid band, and 0.234 on the full grid. This full-grid value is the highest in the model set and remains below its mid-band range. The routing key and Figure 12 use the full 12-point one-step probe grid; we report the mid-band $[25\%, 72.5\%]$ restriction separately as ρ_{mid}. Specifying the grid makes these measurements comparable.

Across six models, full-grid values lie in $[0.10, 0.234]$, with $\Delta\mathcal{A} \leq 0.016$ between UniPC and Euler. HunyuanVideo-1.5 gives 0.215 ± 0.030 over six units. All measured models remain below the 0.5 sentinel. Their positive intermediate overlap leaves the exponent in equation 5 available for calibration. Mid-band values rise from 0.26 on the image DiT through 0.35 on HunyuanVideo-13B to 0.47 on the smallest video model. Excluding the less-aligned early and late directions raises this mean for every model by a model-specific amount.

The equal-magnitude Gram reduction in Section J.4 gives $r_{\mathcal{A}} = 2\log L/\log\big(L[1+\mathcal{A}(L-1)]\big)$. For $\mathcal{A} \in [0.10, 0.234]$ and $L \in [4, 16]$, it spans $[1.30, 1.68]$; mid-band values give $[1.14, 1.41]$. These ranges locate the contribution envelope under equal magnitudes; segment prices are approximately equal in magnitude. Each model uses its own measured, protocol-specific key.

J.7 Factor Ablation and Family Leverage

The FLUX routing audit uses two pricing axes and the 16-value grid $p \in \{1, 1.1, \ldots, 2, 2.5, 3, 4, 6, \infty\}$, giving 32 settings per budget. The records `axes/flux_v2.json` and `axes/flux_v3.json` retain $8, 5, 5, 2$ representative evaluation arms at $k = 12, 14, 18, 22$ after deduplication within each axis. Comparing masks across both axes gives $6, 5, 5, 2$ distinct masks; two masks at $k = 12$ occur on both axes. All 20 representative arms have LPIPS records on each of the 12 calibration prompts.

2×2 ablation. On FLUX's four audited budgets we cross the two axes of Section 4.2: the pricing dimension, pooled against per-unit, and the within-segment exponent, canonical r=2 against the calibration-selected r^*. Cells are verification-pool means of LPIPS-alex over 24 held-out units; r^* comes from the calibration argmin only, and differs by dimension.

k	pool, r=2	pool, r^*	unit, r=2	unit, r^*	Δ exponent	Δ dimension	interaction
12	0.1505	0.1222	0.1505	0.1222	0.0284	0.0000	0.0000
14	0.1005	0.1005	0.1003	0.0814	0.0000	0.0002	0.0189
18	0.0582	0.0449	0.0586	0.0443	0.0132	−0.0005	0.0012
22	0.0234	0.0234	0.0178	0.0178	0.0000	0.0056	0.0000

At k=12, the exponent supplies the entire gain; at k=22, the pricing dimension supplies it. At k=18, the exponent dominates, while the dimension alone slightly worsens LPIPS. At k=14, each factor alone changes LPIPS by at most 0.2%, but together they improve it by 19%. A one-axis search would yield a null result at k=14. Pooled and per-unit pricing select different r^* at three of four budgets, supporting the joint search in equation 7.

Leverage per cell. Leverage $L(k)$ counts distinct masks produced by the family members at budget k on a fixed calibration pool:

pool	budget k	members	$L(k)$
FLUX, 20 units, n=28	6	7	4
FLUX, 20 units, n=28	8	7	5
FLUX, 12 units, sweep	14, 22	16 exponents	1
HYV, 16 units	8	7	7
HYV, 16 units	10	7	7
HYV, 16 units	14	7	6

At L=1, every family member compiles the same schedule, so equation 7 confirms a single candidate. The two L=1 budgets in the fine sweep coincide with the zero-exponent-gain cells above. Reporting L with each family result identifies how many distinct masks calibration must compare.

J.8 Placement Information Captured by the Pooled Schedule

Per-prompt budget sensitivity spans about $5\times$ at k=18.5; see Figure 14. Early-trajectory statistics correlate with optimal tail structure, reaching $|\rho| = 0.808$ at permutation-test $p < 5 \times 10^{-5}$, with more than twenty of 648 tested pairs above $|\rho| = 0.5$. The online placement solve takes 6.6 ms. Using this signal, we evaluated an energy-binned mask bank, chord extrapolation, and a partial-trajectory solver against a prespecified gain criterion. Their Jaccard scores against the oracle mask range from 0.482–0.644. The best reconstruction matches the pooled schedule; only the per-prompt oracle meets the gain criterion. These results support deploying one shared mask per budget as a table lookup. The oracle gain motivates testing predictors based on per-prompt perceptual readouts.

J.9 Reproduction of the Family

Algorithm 1 PropCache: calibrate once, select per budget, compile offline

Require: velocity field $v_\theta(x, \sigma)$; noise nodes $\sigma_0, \ldots, \sigma_n$
Require: inference operator; budget set $\mathcal{K}$; steps $t = 0, \ldots, n-1$
Ensure: precomputed mask table $\{m_k^\star\}_{k\in\mathcal{K}}$

Calibrate: all-compute references on calibration prompts under the inference operator
probe gain profile $A(t)$ and cache-out directions $\hat{d}_s$; apply the overlap verification protocol in Section J.6
Price: $\ell(a,t) \leftarrow A(t)\,|\Delta\sigma_t|\,d(a,t)$ per anchor–step pair (a,t), at pooled and per-unit resolutions
segment-cost matrices $S_r[a,t]$ for the fixed exponent grid $\mathcal{R}$, by prefix sums
for $k \in \mathcal{K}$ **do**
  for $\mathrm{dim} \in \{\mathrm{pool}, \mathrm{unit}\}$, $r \in \mathcal{R}$ **do**
    solve the bottleneck recursion for mask $m_{\mathrm{dim},r}(k)$ with exactly k computed steps
  end for
  $(\mathrm{dim}^*, r^*)(k) \leftarrow \arg\min$ calibration objective equation 7, tie-break smaller r then pooled
  $m_k^\star \leftarrow m_{\mathrm{dim}^*,r^*}(k)$; store the mask and its calibration provenance
end for
return $\{m_k^\star\}_{k\in\mathcal{K}}$; serving is one table lookup per request

The family in equation 5 reconstructs masks directly from the priced matrices. Continuous integration checks byte-identical reconstruction of all member masks across the three models' budget axes. Selection by equation 7 uses one GPU calibration-pool readout per distinct mask; the leverage in Section J.7 bounds this count, which is reduced by collisions among family members. Every PropCache arm enforces n_{calc}=k per unit and uses a same-process dense reference for perceptual evaluation. Budget-inexact arms are labeled with their realized compute and control parameter and reported separately from budget-exact groups. Threshold-swept baselines use the setting nearest the target budget, preferring the side that spends at least as much.

FLOPs accounting. Step caching changes the number of transformer forwards at a fixed per-forward cost. Tables 1, 3, and 13 use analytical FLOPs from `scripts/flops_model.py` and the shipped backbone configurations in `figdata/upstream/configs`. At the frozen generation shape, the script counts attention projections, both attention products, feed-forward layers,

patch embedding, text projection, and the output head at two FLOPs per multiply–accumulate. It multiplies this cost by each arm's forward count and the two classifier-free branches where used. Excluded operations are normalizations, activations, and rotary embeddings, whose elementwise work costs $O(1/d)$ of a projection; timestep and pooled-modulation MLPs, at under 0.02% of the total; and the text encoder and VAE, whose per-request costs are shared across methods.

Per-forward costs are 1.68 P for Wan2.1-T2V-14B and Wan2.2-T2V-A14B at $81 \times 480 \times 832$ with 512 text tokens, 1.25 P for HunyuanVideo-13B at $61 \times 544 \times 960$, and 0.074 P for FLUX.1-dev at 1024^2. Wan2.2 activates one expert per step, using 40 of its 80 blocks per forward. Its cost agrees within 4.3% with the upstream campaign's independent block-equivalent estimate; the build asserts the reported gap. The script also gives 0.28 P for Wan2.1-T2V-1.3B. Table 10 reports forward counts for both models in its block. HunyuanVideo-1.5's campaign leaves the generation shape and padded text length unspecified at the precision required for exact FLOPs.

Figure fidelity. The qualitative grids and page-one strip retain the generated frames' full pixel counts. `scripts/fig_qual_grid.py` and `scripts/fig_hero_flux.py` draw vector layers with matplotlib and embed source-resolution rasters directly, avoiding backend resampling that would remove roughly half the linear detail. Tiles at least three times the print-resolution floor use JPEG quality 95 without chroma subsampling to meet the submission size limit. Compression yields 43 to 51 dB PSNR, compared with 13 to 30 dB between cached and dense samples in Tables 1 and 3. The compression error is one to two orders of magnitude below the cached-versus-dense error. Other tiles, including all video frames, retain their source bytes. `scripts/audit_figures.py` checks PDF pixel counts, effective resolutions, and filters against the generator manifests. Quantitative evaluation uses source outputs.

Repeated baseline arms. BudCache's stage-1 search combines simulated annealing with hill-climbing refinement. Two runs at 14 forwards on Wan2.1-T2V-14B returned distinct, budget-exact 14-step schedules with LPIPS 0.1087 and 0.1261, PSNR 24.1 and 22.6, and SSIM 0.824 and 0.804. Table 1 reports their mean. Every other baseline row in the two video tables represents a single arm at its stated budget. The release includes per-arm records for both schedules.

HunyuanVideo-1.5 baseline correction. The original MagCache and TeaCache campaign used one forward per step with guidance disabled, then evaluated against a separately generated guided dense reference. TeaCache also replayed its trigger on a forward-indexed trace with two entries per step, consuming a threshold calibrated for 49 steps over 99 firings. The corrected block uses the manifest configuration in one process, with MagCache's threshold and window frozen per budget, TeaCache's trigger recalibrated on the conditional stream at the exact forward count, and a dense reference sharing the process and seed.

MagCache's LPIPS improves from 0.5842 to 0.1878 at $k{=}20$ and from 0.5214 to 0.1499 at $k{=}25$. TeaCache improves from 0.7511, 0.6942, and 0.6766 to 0.6493, 0.6195, and 0.5638 at $k{=}14$, 20, and 25. The unaffected MagCache arm at $k{=}14$ reproduces at 0.3359. Corrected MagCache outperforms PropCache at both audited budgets and is bolded in Table 10. Same-process wall-clock measurements are 174.9 s for dense inference and 87.2 s and 69.8 s for the cached arms. These timings determine the reported speedups. The release includes all 156 per-arm records, 48 calibration definitions, and the aggregate under `figdata/upstream/hyv15_fix_timed`.

Table 20 lists each model's calibration and evaluation pools. The Wan2.2 Pareto ladder uses its historical cohort for comparison with the original campaign and is the sole overlapping pool. The transfer result in §5.1 uses the disjoint split listed separately, with evaluation units held out from the solver.

Wan2.1-T2V-14B's 943-prompt suite and 24-unit calibration pool are disjoint by prompt text. Extending the exponent grid updates the candidate list and repeats the same solve; evaluating the resulting masks uses GPU time.

J.10 Analytical Results and Calibrated Quantities

The tables below collect the analytical results with their assumptions and the calibrated quantities with their measurement protocols and values. Together they specify the compiled schedule's inputs.

Table 20: Calibration and evaluation pools per model. A "unit" is one prompt×seed request.

Model	Calibration	Evaluation	Overlap
Wan2.1-T2V-14B	24 units	16 units; 943-prompt suite	none
Wan2.1-T2V-1.3B	6 units	6 held-out units	none
Wan2.2-T2V-A14B	24 traj., 12 prompts	ladder: 16-request cohort	$12/16$
		transfer: 16 unseen-by-solver units	none
HunyuanVideo-13B	2 probe units	16 units	none
HunyuanVideo-1.5	20 units	16 units	none
FLUX.1-dev	12 prompts	88 unseen prompts	none

Established by algebra.

Object	Scope
equation 3, equation 9 and equation 10	First-order propagation at a fixed mask with augmented-state transitions carrying anchor drift. Cached-step contributions superpose in this linearized system.
equation 12, equation 13	Identities for nonzero denominators, separating price scale from fixed-mask geometry and response change across executed masks from individual-response geometry.
equation 11	Exact telescoping identity for augmented transitions. Each cached step contributes one term pairing its Jacobian and running deviation with the anchor Jacobian and stored deviation. Each term is first order in h_j.
equation 14 envelope	For contribution norms $\nu \geq 0$ and a Gram matrix with $G_{ij} \geq 0$, ℓ_2 and ℓ_1 enclose the linearized error. A matching exponent exists in $[1, 2]$ for each segment, uniquely when at least two contributions are nonzero.
equation 15	Exact for $r \geq 1$ and nonzero nonnegative price vectors; it bounds the ratio between two price-based costs.
equation 5 + solver	Budget-exact and acyclic; matrix in $O(n^2)$ by prefix sums, one bottleneck recurrence for every r.

Calibrated by measurement.

Quantity	Protocol and measured value
gain profile $A(t)$	One-step probes on a 12-point grid per model, sampler, and deployment operator, pooled across calibration units. All six schedules decay overall by $\times 11$–$\times 107$; see Table 18. Four profiles decrease strictly; Wan2.1-T2V-14B and HunyuanVideo-1.5 each have a final increase of approximately 8% and 17%, respectively. Placement uses the pooled scalar profile.
price rank validity	Log–log slope 0.76, R^2=0.77 on 784 single-step cache-outs of Wan2.1-T2V-14B; slopes 0.72–0.97 with $R^2 \geq 0.82$ on three further models spanning scale, MoE routing, and a single-forward backbone. Calibration uses relative price rankings.
pooling across prompts	Per-prompt log-profile correlations of 0.94–0.98 and near-constant scale variation of $\pm 22\%$ support one pooled profile per deployment configuration.
overlap key $\mathcal{A}$	Mean pairwise cosine of one-step terminal directions: $[0.10, 0.234]$ on the full 12-point grid across six DiTs, 0.26–0.47 on the mid-band restriction. The interference-pair cosine has median 0.448. The full-grid mean supplies the routing gate.
two-factor split of equation 13	Per pair on the 12 interference pairs of Wan2.1-T2V-1.3B: median geometry 1.165, response ratio 1.625, composite 1.901; the identity holds to 8×10^{-10} per pair.
geometric factor on a pair	Measured slack 1.15–1.19 on the mid-band pairs against the equal-norm bound 1.15–1.20 at each pair's own cosine. The early pair has slack 1.03–1.08 and equal-norm bound 1.37–1.40.
response ratio	$\|\delta_s + \delta_t\| / \|\delta_{s,t}\|$: 1.00–1.01 on the early pair and 1.5–1.7 on the ten mid-band pairs, comparing the individual and joint interventions.
composition exponent $\hat{r}$	Two-window composition fit $\hat{r} = 3.0$, interval $[2.9, 3.1]$, in-sample residual $\leq 4.3\%$, held-out $\leq 2.8\%$; see Section J.5. The deployed exponent is selected per budget by equation 7.
price dispersion	Pricing dimension acts through relative price variation because the bottleneck solve is invariant to global rescaling; see Section J.7.
per-budget selection	10 tiers on one image and two video DiTs, a prespecified criterion, held-out verification pools, and equal budgets; win rates reported per tier.