CLOSED-FORM LEAST-SQUARES FORECASTING FOR FAST DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion transformers spend most of their inference budget re-evaluating a network whose outputs change slowly along the sampling trajectory. Cache-based accelerators exploit this redundancy by skipping selected steps, and their quality is governed by two levers—which steps to compute and how to fill in the skipped ones—that have mostly been studied in isolation. We build a framework that exposes both at once and use it as a controlled attribution study. Our predictor treats the sequence of whole-transformer outputs as the observable of a discrete-time system and forecasts a skipped output as a linear functional of previously computed outputs, fitted either per image by global least squares or, once for all images, by closed-form calibration on a handful of prompts requiring no gradient training. The attribution result is unambiguous and survives every control we apply: *step selection dominates predictor design*, and per-image adaptive gating is worse than committing to a schedule even at a lower skip rate. Repeating the whole protocol on SD3.5-Medium, a 2.5B MMDiT-X model with flow matching and a different text stack, shows the same ordering and a better operating point: $3.39\times$ wall-clock on MJHQ-200 at 0.34 CLIP below the uncached model with 97% of its sharpness, and CLIP at or above the uncached model at a 30-step budget and at 768px. On PixArt-α the method reaches 2.3–$2.9\times$ at alignment matching the uncached model on two benchmarks.

1 INTRODUCTION

Diffusion transformers define the current state of the art in text-to-image generation (Peebles & Xie, 2023; Chen et al., 2024; Rombach et al., 2022; Esser et al., 2024), but they are expensive to run: each image requires tens of sequential evaluations of a multi-billion-parameter network over thousands of tokens (Ho et al., 2020; Song et al., 2021). At deployment scale, this sequential cost, not the model's parameter count, is what sets latency and energy, so reducing the number of evaluations is a first-order practical problem.

Existing remedies fall into two groups, each with a structural limit. *Sampler-side* methods change the trajectory itself, by distilling it into fewer steps (Salimans & Ho, 2022; Song et al., 2023; Liu et al., 2023) or integrating it with higher-order solvers (Lu et al., 2022; Karras et al., 2022); they are powerful but change what a step means and interact subtly with classifier-free guidance (Dhariwal & Nichol, 2021). *Caching* methods leave both model and sampler untouched and instead skip selected evaluations, reusing or extrapolating what has already been computed (Ma et al., 2024b; Zou et al., 2025; Liu et al., 2025c;a); this makes them training-free and composable with the methods above. Within this second group, however, quality is set by two coupled choices—which steps to compute, and how to fill in the skipped ones—and published methods tune one while freezing the other by hand (Shen et al., 2026; Cui et al., 2026). The consequence is not that any method is wrong, but that the field has no measurement of which axis actually governs the speed–quality trade-off, and therefore no principled basis for choosing where to spend effort.

This paper supplies that measurement. We view the sequence of whole-transformer outputs as the observable of a discrete-time system and forecast skipped outputs by a linear functional of the computed ones, with weights obtained *in closed form*—either per image by a global least-squares fit in the step index, in the spirit of the Koopman/DMD line on data-driven linear representations (Schmid, 2010; Tu et al., 2014; Brunton et al., 2022), or once for all images from a small calibration set. The

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.

closed form is what makes the study possible: no gradient training and no hyperparameters means the predictor is a fixed instrument, so the schedule can be varied against it cleanly.

Our contributions are as follows. **(1)** A closed-form forecasting method whose skipped steps are filled by a least-squares predictor calibrated once on a handful of prompts. **(2)** A controlled attribution study showing that step selection dominates predictor design, and that per-image adaptive gating is dominated by a fixed schedule. **(3)** Cross-architecture validation: the same procedure on SD3.5-Medium (MMDiT-X, 2.5B) gives $3.39\times$ at 97% of the reference sharpness, and exceeds the uncached model's alignment at a 30-step budget and at 768px.

The paper is organized as follows. Section 2 reviews the two families and the linear-representation work our estimator draws on. Section 3 states the problem, the forecasting method, its one-shot calibration and the schedule design. Section 3 closes with what can be said about the forecast and the measured envelope of its error bound. Section 4 reports the benchmark results, the generality probes, the cross-architecture validation and the attribution study. Section 5 discusses scope and concludes.

2 Related Work

Related work comprises three strands: **feature caching for diffusion models**, **data-driven linear representations of nonlinear dynamics**, and **faster sampling without caching**. On the first strand, DeepCache (Ma et al., 2024b) reuses U-Net features on a fixed interval; ToCa (Zou et al., 2025) selects which tokens to cache; TGATE (Liu et al., 2025b) caches attention outputs once they converge; TeaCache (Liu et al., 2025a) gates each step on a timestep-embedding statistic; and FBCache recomputes the first block at every step and caches the residual of the rest, preserving detail by construction (it ships with the Diffusers library (von Platen et al., 2024) rather than a publication, so we treat it as an engineering baseline). TaylorSeer (Liu et al., 2025c) replaces reuse by Taylor extrapolation, and a recent group shapes the cache's temporal structure through quantisation-aware schedules (Liu et al., 2025d), trajectory-wide reuse distribution (Chu et al., 2025) or frequency-aware video caching (Lyu et al., 2025). Closest to us, L^2P (Shen et al., 2026) observes that forecasting caches reduce to fixed linear combinations of past features and fits per-timestep weights by *gradient descent*, while DPCache (Cui et al., 2026) selects computed steps by dynamic programming and fills the rest by Taylor prediction. We estimate the same kind of linear predictor in *closed form*, with no optimisation and no hyperparameters. Our calibration set is held fixed while the schedule is varied, which is what enables the attribution study of Section 4.5; both estimators are cheap, and the difference is the form of the estimator (Appendix A). On the second strand, the Koopman operator framework lifts a nonlinear flow into a space of observables on which the dynamics act linearly (Brunton et al., 2022), and dynamic mode decomposition estimates the resulting operator from snapshot data by least squares (Schmid, 2010; Tu et al., 2014; Williams et al., 2015); learned-observable variants and operator-learning approaches such as Fourier neural operators (Li et al., 2021) and DeepONet (Lu et al., 2021) extend the same idea to deep models. We borrow the *estimation principle* of this line—a global least-squares fit of a linear input–output map—and apply it to the sampling trajectory of a diffusion transformer, deliberately *not* constructing an operator, learning observables, or performing modal decomposition, since the trajectory window is short and the observable is the full transformer output. On the third strand, distillation and solver-based methods reduce the number of steps itself (Salimans & Ho, 2022; Song et al., 2023; Liu et al., 2023) or use higher-order ODE solvers (Lu et al., 2022; Karras et al., 2022); these are complementary to caching—they change what a step means, whereas we skip steps of a given solver—so we evaluate against caching baselines at a fixed solver (20 DDIM (Song et al., 2021) steps in all experiments).

3 Method

3.1 Problem setup

Let a pretrained diffusion transformer ϵ_θ define a sampling trajectory $x_0, x_1, \ldots, x_N$ produced by a fixed solver, e.g. DDIM updates

$$x_{t-1} = \alpha_t x_t + \beta_t f_t, \qquad f_t = \epsilon_\theta(x_t, t, c), \tag{1}$$

where c is the prompt conditioning and α_t, β_t are solver coefficients. A caching accelerator computes f_t at a subset $\mathcal{S} \subseteq \{0, \dots, N-1\}$ of *full* steps (t indexes the solver iteration in execution order) and substitutes an approximation $\hat{f}_t$ elsewhere. The quality of the result depends on $\mathcal{S}$ (the schedule) and on the rule that produces $\hat{f}_t$ (the predictor), and the achieved speedup is governed by $|\mathcal{S}|$ plus per-step overhead.

Write $\mathcal{H} = \{(s_i, f_{s_i})\}_{i=1}^{m}$ for the outputs already computed when the sampler reaches step t. We consider predictors that are linear in the cached history, $\hat{f}_t = \sum_{i=1}^{m} w_i(t)\, f_{s_i}$, with weights depending only on the step indices and not on the image. This class covers verbatim reuse ($w_m = 1$, as in DeepCache and FBCache), fixed-window extrapolation, and the learned temporal weights of L^{2}P. Our question is how to choose w and $\mathcal{S}$.

3.2 Forecasting the denoising trajectory

We treat f_t as an observable of the sampling dynamics and assume that, over the window we need, it is well approximated by a smooth function of the continuous step index: $f(u) \in \mathbb{R}^d$ with $f(s_i) = f_{s_i}$. Approximating f by a degree-q polynomial in u and evaluating it at $u = t$ gives weights

$$w(t) \;=\; \Phi \left(\Phi^\top \Phi\right)^\dagger \Phi_t, \qquad \Phi = \left[\, u_i^k \,\right]_{i,k}, \quad \Phi_t = \left[\, u_t^k \,\right]_k, \tag{2}$$

where $w(t) \in \mathbb{R}^m$ is the weight vector applied to the cached outputs, the basis is centred on the first computed step ($u_i = s_i - s_1$, $u_t = t - s_1$), and $(\cdot)^\dagger$ is the Moore–Penrose pseudo-inverse computed by an SVD-based solve. Equation 2 is a global fit: all cached outputs contribute, and the weights are the *leverage* of each history point for the extrapolation at t. We use $q = 1$ (affine) throughout unless stated otherwise.

The choice of a global least-squares fit rather than finite differences is deliberate: on an irregular grid—which aggressive schedules produce—a finite-difference error constant grows with the ratio of extrapolation distance to local spacing, whereas the least-squares fit minimizes the residual over the whole window. On our schedules the two attain nearly the same per-step error (Appendix E); we keep the least-squares form for its spacing-independent worst-case bound (Section 3.5).

The per-step procedure is two lines (Appendix Table 5): a full step is computed by the unmodified model and appended to the history; a skipped step costs one weighted sum over at most m cached tensors and returns a tensor with the transformer output's shape and semantics, so the solver is untouched. With K full and $N - K$ predicted steps the ideal speedup is $N/(K + \kappa(N - K))$, where κ is the prediction cost of a skipped step relative to a full forward pass; the measured speedup sits 10–25% below N/K, the gap growing with K and attributable to per-step costs that skipping does not remove. Because wall-clock time also varies between batches (Appendix C), we compare methods only within a batch and report s/img alongside the ratio.

3.3 One-shot closed-form calibration

The weights in Eq. 2 are estimated from a single trajectory, where only m outputs are available before the first prediction—typically two to four. Estimating m weights from m noisy observations of one trajectory is a small-sample problem; pooling trajectories (Eq. 3 below) reduces estimator variance (Section 3.5) and improves sharpness at essentially unchanged alignment (Section 4.5).

We therefore estimate the weights of the same affine form *across* trajectories, in closed form: Eq. 2 extrapolates along the step index, Eq. 3 fits that form to snapshots pooled over prompts. Because the weights depend only on the step indices and on the length of the available history, we parameterize them by history length, $w^{(m)} \in \mathbb{R}^m$, and fit them by ordinary least squares on a calibration set of P prompts whose trajectories are collected by running the unmodified model:

$$w^{(m)} \;=\; \arg\min_{w} \sum_{p=1}^{P} \sum_{t \,\in\, \mathcal{T}_m} \left\| \, f_t^{(p)} - \textstyle\sum_{i=1}^{m} w_i f_{s_i(t)}^{(p)} \, \right\|_2^2, \tag{3}$$

where $\mathcal{T}_m$ collects the steps at which exactly m outputs are available and $s_i(t)$ are their indices. Which steps belong to $\mathcal{T}_m$ is fixed by the schedule, so steps that see the same number of computed outputs—including consecutive steps inside one skip run—share the same weights. Equation 3 is

an $m \times m$ normal-equation solve over pooled snapshots: with $P = 20$ prompts it takes seconds on a single GPU, involves no gradient training and does not modify the pretrained weights. The same calibration set is used for every *schedule* and for every computed-step budget K at a given sampler budget N, and—because the weights are fitted on the sampler's trajectory rather than on a schedule—they transfer across the benchmarks we test without refitting (Section 4). Changing the sampler budget N re-discretizes the trajectory and requires re-solving Eq. 3 (Section 4.3).

3.4 SCHEDULE DESIGN

Given a predictor, the remaining decision is the set $\mathcal{S}$ of computed steps. Two quantities constrain it. First, the number $K = |\mathcal{S}|$ sets the speedup. Second, the predictor is used at step t only after at least two outputs exist, and its error grows with the distance to the most recent computed step; we therefore bound the maximal gap

$$g(\mathcal{S}) \;=\; \max_{t \notin \mathcal{S}} \big(t - \max\{s \in \mathcal{S} : s < t\}\big) \;\le\; g_{\max}. \tag{4}$$

Our schedules place K steps to satisfy Eq. 4 with $g_{\max} \approx 5$ and otherwise spread the remaining budget evenly; anchoring the first three steps and covering the full trajectory range gave the best alignment–speed trade-off. All schedules are listed in Appendix Table 33, and Appendix G states the empirically derived four-step design rule (G1)–(G4) and verifies it for every schedule used here.

The gap bound has a spectral reading, and it is the reason we cap it: the spectral radius of the companion matrix induced by the fitted weights is 0.97–1.03 for histories of length 3 to 7—numerically neutral—but rises to 1.37 at length 8, where the fit begins to extrapolate rather than interpolate (Appendix Table 30). On SD3.5 the same quantity stays neutral even at length 8 (1.001), so there the bound is conservative (Appendix D.4).

3.5 WHAT CAN BE SAID ABOUT THE FORECAST

We close this section with four things that can be said about the forecast without assuming anything about the network that we do not also measure. Throughout we fix $q = 1$, the affine model used in every experiment, and write $u_1 < \cdots < u_m$ for the step indices of the m outputs available when a prediction is needed, centred as $u_i = s_i - s_1$, with $\bar{u} = \frac{1}{m}\sum_i u_i$ and $S_2 = \sum_i (u_i - \bar{u})^2$.

The fit is exact where it matters, and stays well conditioned where it is hard. If the observable is affine in the step index, $f(u) = a + bu$, then $\hat{f}_t = f(t)$ for every t, whatever the spacing of the computed steps. A one-step finite-difference forecast of the same order is also exact on that class; the two differ once the observable is not exactly affine, where the finite-difference truncation error carries the factor $(t - u_m)/(u_m - u_{m-1})$, which grows without bound when the last two computed steps are close together while the step to be predicted is far ahead, whereas the least-squares residual is bounded by the leverage of Eq. 5. This is the practical reason we keep the least-squares form on the irregular grids that aggressive schedules produce, rather than a local finite-difference scheme.

The weights sum to one, and the schedule controls how sensitive they are. The weights in Eq. 2 satisfy $\sum_i w_i(t) = 1$ for every t, so a constant observable is forecast without bias, and

$$\|w(t)\|_1 \;\le\; 1 + \frac{|u_t - \bar{u}|}{S_2} \sum_{i=1}^{m} |u_i - \bar{u}|. \tag{5}$$

So a perturbation of the cached outputs of size δ perturbs the forecast by at most $\big(1 + |u_t - \bar{u}| \sum_i |u_i - \bar{u}|/S_2\big)\delta$: the amplification grows with the distance between the predicted step and the centre of the computed window, and shrinks as the computed steps spread out. Bounding the largest gap is therefore the direct way to control this factor, not a convenience.

Pooling the calibration set lowers variance without introducing bias. Let $\hat{w}^{(m)}_{\text{img}}$ be the weights estimated from a single trajectory and $\hat{w}^{(m)}_{\text{cal}}$ those of Eq. 3 estimated from P trajectories. If the deviation of a trajectory from the fitted model is independent across prompts and has variance σ^2—an idealisation, since all trajectories come from the same network and so share model bias—then

both estimators are unbiased for the same population weights and $\mathrm{Var}(\hat{w}_{\mathrm{cal}}^{(m)}) = P^{-1}\,\mathrm{Var}(\hat{w}_{\mathrm{img}}^{(m)})$. This bounds the variance of the fitted weights and so constrains the *alignment* side of the trade-off; the sharpness side is governed by schedule geometry, which Section 4.5 characterizes empirically.

Over a skip run, errors stay within an envelope we can now measure. If each prediction introduces an error of at most ϵ in the network output, and a perturbation δ at step t produces a perturbation of at most $\rho\|\delta\|$ at step $t-1$, then after k consecutive skipped steps the deviation is at most $\epsilon(\rho^k - 1)/(\rho - 1)$ for $\rho > 1$ and at most ϵk for $\rho \leq 1$. Rather than assume a value for ρ, we measured it: we perturb one block's output at step t by δ (0.01 of its norm) and record the Frobenius-norm magnification of that block's output at $t{+}1$ on real trajectories (Appendix D.5). On PixArt-α the per-step magnification over 12 prompts and six blocks has median 0.45 (75% of samples contract) with a bounded worst case of 3.6; on SD3.5 all 180 measurements contract, median 0.177, maximum 0.405. A skipped step therefore usually *damps* the forecast error, which makes the bound above a conservative envelope: a run of length k multiplies the accumulated deviation by at most 3.6^k in the PixArt worst case and by at most $0.405^k < 1$ in every SD3.5 sample, whereas an adaptive gate can produce long runs that no gap cap constrains.

Propositions 1 and 2 describe the weights of Eq. 2, the uncalibrated polynomial predictor; the calibrated weights of Eq. 3 come from a different estimator, but in practice they nearly satisfy the same property—summed over the history they give $\mathbf{1}^\top w^{(m)}$ between 0.97 and 1.00 for $m \leq 6$, so the constant-observable bias stays below 3%. Together these locate the cost of the method: the forecast is exact on the component of the trajectory it is designed for, its sensitivity is a function of the schedule rather than of the predictor, and its error over a run decays rather than grows on both backbones we test. These statements are derived under idealized affine dynamics: they are analytical grounding for the design rather than exact performance predictions, and Section 4 tests the design they motivate.

4 Experiments

4.1 Setup

Model and data. All experiments use PixArt-α (Chen et al., 2024) at 512×512 resolution with $N = 20$ DDIM steps and classifier-free guidance (Dhariwal & Nichol, 2021) scale 4.5, on a single NVIDIA GeForce RTX 4080 Laptop GPU (12 GB). Prompt embeddings are precomputed with the frozen T5 encoder. We evaluate on two established text-to-image benchmarks: **COCO val2017** (Lin et al., 2014) and **MJHQ-30K** (Li et al., 2024) (a 30,000-image, ten-category benchmark with reference images and prompts). Both use 200 prompts sampled uniformly over categories and a fixed seed.

4.1.1 Metrics

We report wall-clock speedup over the uncached model, CLIP score (Radford et al., 2021) between generated images and their prompts, and image *sharpness*, the mean variance of the Laplacian of the grayscale image. Sharpness is included because CLIP is largely insensitive to blur: a $2\times$ sharpness difference can coincide with a 0.1 CLIP difference, so alignment alone would let a visibly smoothed method pass as lossless. It measures high-frequency energy of either kind, so a value *above* the uncached reference is a warning sign, not an improvement; where this happens our CLIP and ImageReward columns show the corresponding loss. FID is reported for completeness; however, at $n = 200$ its estimator variance dominates the method-level differences, precluding statistically meaningful comparisons. Two noise scales calibrate the claims below: the same configuration run twice differs by less than 0.05 CLIP, and across three 200-prompt seed runs the per-method standard deviation is 0.05–0.12 (Appendix D), so we read differences below ≈ 0.25 as noise-level. We additionally report *ImageReward* (Xu et al., 2023), a learned human-preference model, on the same 200 MJHQ prompts; it correlates with CLIP at $\rho{=}0.89$ over our seven PixArt method points (a wide CI of $[0.40, 1.00]$, so confirmatory rather than primary) and $\rho{=}0.94$ over the ten SD3.5 points.

Baselines. We compare with seven accelerators applied to the same model and solver: FBCache, TaylorSeer (Liu et al., 2025c), TeaCache (Liu et al., 2025a), TGATE (Liu et al., 2025b),

Table 1: Main results on PixArt-α, 512px, 20 DDIM steps, CFG 4.5, 200 prompts per benchmark. Speedup and s/img are relative to the uncached model of the same column; CLIP and sharpness are means over the 200 images (CLIP: higher is better; sharpness: closer to the uncached reference is better). Each row is the single "seed 0" run; three-seed means and per-seed spreads are in Appendix Table 23, which also reports FID. Bold marks the best value per column among the baselines above the midrule. Every row is timed in one batch, so the speeds are comparable across rows.

	COCO val2017				MJHQ-30K			
Method	s/img	Speedup ↑	CLIP ↑	Sharpn. ↑	s/img	Speedup ↑	CLIP ↑	Sharpn. ↑
Uncached (20 steps)	8.053	1.00×	30.00	782.3	7.968	1.00×	30.18	816.5
TGATE (Liu et al., 2025b)	7.718	1.04×	30.33	731.5	7.686	1.04×	30.61	700.6
TeaCache (Liu et al., 2025a)	5.464	1.47×	30.31	113.8	5.578	1.43×	30.18	816.5
ToCa (Zou et al., 2025)	5.027	1.60×	29.31	286.1	4.882	1.63×	28.87	303.2
TaylorSeer (Liu et al., 2025c)	4.405	1.83×	29.92	78.8	4.663	1.71×	29.10	89.5
FBCache	4.220	1.91×	**30.41**	365.4	4.288	1.86×	**30.80**	412.6
DPCache (Cui et al., 2026)	3.955	2.04×	26.23	469.1	3.916	2.03×	22.52	679.6
L^2P (Shen et al., 2026)	**3.165**	**2.54**×	29.83	170.9	**3.156**	**2.52**×	29.76	182.1
Ours (8 steps)	3.516	2.29×	30.07	480.2	3.506	2.27×	30.21	498.0
Ours (7 steps)	3.131	2.57×	30.16	225.8	3.165	2.52×	30.25	233.8
Ours (6 steps)	2.751	2.93×	30.27	124.4	2.782	2.86×	30.12	126.3

ToCa (Zou et al., 2025), L^2P (Shen et al., 2026) and DPCache (Cui et al., 2026). Each baseline is configured at its recommended operating point; where a threshold controls speedup we report the best quality at or above 1.5×, and on MJHQ-30K some thresholds admit no such point. Every method is timed in a single batch: repeated timings of one configuration on a laptop GPU can differ by up to 25%, enough to invert the ranking of two methods whose true latencies differ by 0.2 s. ToCa ships caching for the non-MS checkpoints only, so running the authors' cache functions on the present block verifies the graft to be bit-exact with caching off (Appendix F treats L2C the same way).

4.2 Main results

Table 1 reports both benchmarks side by side; the alignment–speed and sharpness–speed planes are plotted in Appendix Figure 5.

Three patterns hold on both benchmarks. First, ours is the fastest method that matches the uncached model in alignment: 2.57× at CLIP 30.16 on COCO and 2.52× at 30.25 on MJHQ-30K, against 30.00 and 30.18 uncached, and the 6-step point is faster again (2.93× and 2.86×) at the cost of sharpness. Second, the strongest cache-reuse baseline is ToCa, at 1.60× and CLIP 29.31 on COCO: it spends no extra generations either, yet it is beaten on both axes by every one of our points. Third, the learned-weight baselines do not close the gap: DPCache collapses in alignment (26.23 and 22.52) at comparable speed, L^2P is the fastest baseline but is beaten on alignment on both benchmarks, and the baselines that do reach high alignment (TGATE, TeaCache) do so by computing almost everything, TeaCache falling to 1.43× on MJHQ-30K.

On PixArt, FBCache is the strongest of the baselines, and at matched compute ours is ahead of it: every one of our operating points is faster (2.75–3.52 versus 4.22 s per image on COCO) and sharper (480.2 versus 365.4 at 8 steps). FBCache attains a higher CLIP score (30.41 versus 30.07, at 1.91× versus 2.29×); the human-preference measurement does not distinguish it from our 7/8-step points. Section 4.5 traces its sharpness to recomputing a block at *every* step rather than to its interpolation rule. A third benchmark, PartiPrompts (Yu et al., 2022), confirms this ranking (Appendix D): ours is fastest and nominally best aligned, within the noise band.

Human-preference validation. CLIP measures alignment, not preference, so we also score every method on the same 200 MJHQ prompts with ImageReward (Xu et al., 2023): at our headline 7/8-step points it is -1.58 ± 0.96 and -1.57 ± 0.95, indistinguishable from the uncached model

(-1.52 ± 1.01) and from FBCache (-1.47 ± 1.03) while being $2.5\times$ faster than the uncached model and $1.35\times$ faster, and the two axes rank the methods identically (ρ=0.89, CI$[0.40, 1.00]$). It is a seven-point correlation with a wide interval, hence confirmatory; on SD3.5 the check spans ten points and agrees more tightly (ρ=0.94), but there preference penalises the tighter budgets more than CLIP does (Appendix D.4).

Table 2: Human-preference validation on a 200-prompt MJHQ subset (PixArt-α). ImageReward is mean$\pm$std over the 200 images (higher is better) and correlates with CLIP at ρ=0.89; the "aggressive" row is the speed extreme, the headline rows the 7/8-step points. Latencies are the same-batch timings of Table 1; †that row is a different operating point, timed separately.

Method	s/img	CLIP ↑	ImageReward ↑
Uncached (20 steps)	7.97	30.18	-1.52 ± 1.01
FBCache	4.29	30.80	-1.47 ± 1.03
TGATE	7.69	30.61	-1.56 ± 0.95
TeaCache	5.58	30.18	-1.52 ± 1.01
TaylorSeer	4.66	29.10	-1.76 ± 0.79
L^2P	3.16	29.76	-1.67 ± 0.88
Ours (aggressive, skip 0.7)†	2.69	28.77	-1.86 ± 0.69
Ours (7 steps)	3.17	30.25	-1.58 ± 0.96
Ours (8 steps)	3.51	30.21	-1.57 ± 0.95

4.3 Generality: other budgets, resolutions and solvers

Everything above fixes one sampler budget (20 DDIM steps), one resolution and one model. We probe each axis on PixArt-α (Appendix Table 8); the same three probes are repeated on SD3.5 in Appendix Table 9b. (i) The operator is fitted to a *sampler budget*, not to the model alone: moving from 20 to 30 steps without re-fitting costs 1.7 alignment points and inflates the Laplacian statistic to 2274 (high-frequency artefacts), whereas re-fitting the same closed form on 20 prompts of the new budget restores them, leaving a 30-step point ($3.20\times$) that is both the fastest and 0.9 above the uncached 30-step baseline; the SD3.5 analogue is $3.24\times$ and *above* its uncached baseline. (ii) Resolution is not special: at 768px the method keeps $2.63\times$ with alignment within noise of the uncached model and unchanged position relative to FBCache ($3.68\times$, CLIP 31.35 vs. 31.39 on SD3.5). (iii) Caching and better solvers act on different axes and compose: DPM-Solver++ at 10 steps is already $1.89\times$, and putting our cache on top reaches $2.99\times$, 0.21 below the solver alone and 0.67 above the 20-step uncached baseline.

4.4 Cross-architecture validation on SD3.5-Medium (MMDiT)

Every result so far comes from one backbone, PixArt-α (0.6B, single-stream DiT, DDIM). If the advice "spend your effort on step selection, not on the predictor" were an artefact of that backbone it would be worth little, so we repeat the protocol on **SD3.5-Medium** (Esser et al., 2024), a 2.5B MMDiT-X model with dual-stream joint attention, flow matching, a different text stack (CLIP-L+CLIP-G+T5-XXL) and a different VAE. The operator, the schedule construction and the calibration are used *unchanged*; only the 20-prompt calibration set is re-collected on this model.

The released 4-bit (`q4_k_s`) weights are used for this experiment, allowing a 2.5B MMDiT to run on the same 12 GB card as the PixArt experiments; the cached and uncached configurations share these identical weights, so quantization is common to both rows and cancels in the difference we report.

The method transfers across backbones: the backbone change reduces the cost while preserving the benefit. With 8 of the 28 transformer evaluations we run $3.39\times$ faster on MJHQ-200 and lose 0.34 CLIP (paired over 200 images, $t = -3.0$) while keeping 97% of the reference sharpness; FBCache, $1.8\times$ slower, keeps 60%. The comparison is paired within each backbone: cached and uncached rows share identical weights, so the CLIP difference isolates the caching effect. What it establishes is a cross-backbone pattern: the eight-step budget that costs 55–93% of the sharpness on PixArt's

Table 3: Cross-architecture validation on SD3.5-Medium (MMDiT-X), 512px, 200 MJHQ-30K prompts, CFG 4.5, 28-step flow-matching trajectory (against 20 DDIM steps on PixArt, so step counts and speedups are not comparable across backbones). "Ours" computes 8, 7 or 6 of the 28 steps and forecasts the rest with the same closed-form operator. All rows share the 4-bit weights, so the CLIP difference isolates caching.

Method	Computed	s/img	Speedup ↑	CLIP ↑	Sharpn. ↑
Uncached	28/28	3.78	1.00×	**32.15**	818.9
FBCache	28/28	2.01	1.88×	31.09	491.7
L^2P	11/28	1.56	2.42×	31.99	794.8
Ours (8 steps)	8/28	1.12	3.39×	31.81	795.4
Ours (7 steps)	7/28	0.98	3.88×	31.29	922.5
Ours (6 steps)	6/28	**0.83**	**4.53**×	30.73	1107.0



Figure 1: SD3.5-Medium (MMDiT), 512px: uncached (28 evaluations, top) against ours with 8 computed steps (bottom), same prompts and seeds—consistent with the 0.34 CLIP gap and the 97% retained sharpness of Table 3.

20-step solver costs 3% here, and on PixArt the re-calibrated 30-step point reaches the uncached alignment, whereas on SD3.5 the 30-step and 768px points *exceed* it (Appendix Table 9b). Figure 1 shows side-by-side samples.

The attribution transfers. Appendix Table 9a tests our central claim on SD3.5 and reproduces the PixArt conclusion at matched cost: the fixed 7-step schedule and the gate at τ=0.10 both run at 0.98 s per image, and the schedule loses 0.06 CLIP while the gate loses 0.45; at τ=0.02 the gate is slower than the *uncached* model because it elects to compute nearly every step. The predictor ablation tells the same story from the other side: at a fixed 8-step schedule, verbatim reuse, the affine fit, recency weighting, a predictor–corrector blend and a second-order solver update are all within 0.10 CLIP of one another and *none* of the differences is significant (paired t-tests, p=0.25–0.70); only the quadratic fit is clearly harmful (−2.11 CLIP). Step selection thus matters as much as on PixArt while the predictor matters even less (Appendix D.4 gives the full table).

Appendix D.4 reports the *complete* replication on this backbone: six of the seven baselines of Table 1 on the MMDiT block, both prompt sets, the human-preference metric, the second-order-solver probe, three seeds, a calibration-size sweep, and the weight-variance, spectrum and Lipschitz analyses of Appendix D.5.

Table 4: Step selection at matched compute on COCO val2017, fixed 7-step schedule $[0, 1, 2, 5, 9, 14, 18]$ (skip rate 0.65), 200 prompts. Fixed-schedule rows are means$\pm$std over three seeds (n=3); the adaptive rows run at their own operating point and are not sparsity-matched (seed 0). Between the fixed rows CLIP differs by Δ=0.18—about three times the seed spread—and sharpness by $\approx 2\times$. Predictor and gate panels: Appendix Tables 6, 7.

Step selection	Skip step uses	s/img	CLIP ↑	Sharp. ↑
Fixed schedule	verbatim reuse	3.124	$\mathbf{30.38 \pm 0.03}$	120.9 ± 16.1
Fixed schedule	least-squares forecast	3.126	30.21 ± 0.06	$\mathbf{241.5 \pm 27.5}$
Adaptive gate (τ=0.05)	least-squares forecast	2.590	20.34 ± 0.06	72.8 ± 16.6
Adaptive gate (1-Wasserstein)	verbatim reuse	4.006	30.05	148.6

4.5 Ablations

Step selection versus predictor design. Table 4 and Appendix Tables 6–7 collect the controls that carry the attribution claim. Panel (a) holds the 7-step schedule and the sparsity fixed and changes only what fills a skipped step: verbatim reuse is 0.002 s per image cheaper and scores 0.18 *higher* in alignment (mean over three seeds, inside our cross-seed band) while being about half as sharp, so the predictor provides almost all of the sharpness but no alignment. The predictor sweep (panel b) finds the affine fit the best of six designs we tried beyond it —quadratic, clean-signal, recency, predictor–corrector, gating and a depth–time hybrid— and the gate sweep (panel c) lands 7.7–10.4 alignment points below a fixed schedule even at a *lower* skip rate. Panel (c) includes adaptive gating results where the gate is uncapped, the regime analyzed in Proposition 4; these rows are not sparsity-matched with the fixed schedules. A gate carrying the same gap cap as our schedules, at matched cost on SD3.5, is still dominated (Appendix Table 9a). The effect sizes are large and seed-stable: between the two fixed rows of panel (a), CLIP Δ=0.18 (about three times the seed-to-seed spread) and sharpness $\approx 2\times$, over three seeds. The same ordering reappears on SD3.5 (Section 4.4).

5 Discussion and conclusion

Where the compute goes. Our ablations separate the two quality axes at matched compute. Alignment is governed by *which and how many* steps are computed: at a fixed schedule, verbatim reuse moves alignment by 0.18 (inside the noise band) while an adaptive gate loses 7–10 points. Sharpness is governed by the predictor: at the same schedule the forecast doubles sharpness for 0.002 s per image, and pooling the calibration set multiplies it by a further 2.0–$3.5\times$. The attribution is consistent: compute is best spent on the step schedule, with the predictor kept simple and well estimated. On SD3.5 step selection dominates while the predictor matters less: verbatim reuse matches the affine forecast, and one calibration trajectory suffices.

Scope. We report three operating points (6, 7, 8 computed steps) because the best schedule depends on the step budget; at 8 steps the sharpness loss caching methods incur narrows to about $1.7\times$ from the $2\times$ or more of the family. Differences inside our noise band are read as such: on PixArt, alignment sits within the seed-to-seed spread of Appendix Table 23. One scope limitation: the SD3.5 experiments use the 4-bit release because a 2.5B MMDiT in `bf16` does not fit the 12 GB card; its absolute numbers are quantized, while cached vs. uncached rows remain comparable since the weights are shared.

Conclusion

We studied the two design axes of cache-based diffusion acceleration under matched compute. A whole-transformer least-squares forecast estimates the trajectory's linear operator in closed form; pooling a small calibration set improves sharpness up to $3.5\times$ at unchanged alignment. On PixArt-α the method runs 2.3–$2.9\times$ faster at matching alignment on two benchmarks, and on SD3.5-Medium $3.39\times$ at 97% of the reference sharpness. Step selection dominates predictor design, and per-image adaptive gating is worse than a fixed, gap-bounded schedule at matched cost (Appendix Table 9a); this attribution is a property of the method family, not of a single backbone.

AI USE STATEMENT

Large language models (LLMs) were employed exclusively to improve the quality of written English and for stylistic refinement, with attentive supervision by the authors. No scientific content, experimental design, data analysis, or any intellectual contribution were generated through LLMs, nor did such assistance influence any reported results.

ETHICS STATEMENT

We adhere to the ICLR Code of Ethics throughout this work. The study uses two public benchmarks (COCO val2017 and MJHQ-30K) and one public PartiPrompts set that contain no personal or human-subject data, and a public, openly licensed model checkpoint; we release no new dataset or model, so no additional data-release concerns arise. The method is inference-only and changes no model weights. The main potential harm is the one shared by all text-to-image generators—the possibility of generating misleading or biased imagery—and because our method consumes the same prompts and produces the same distribution as the uncached model, it neither adds a new failure mode nor changes model behaviour outside the caching effect we measure; indeed, faster generation can amplify any such harm, which is an argument for measuring the quality cost honestly, as we attempt to do. The authors declare no conflicts of interest and no external sponsorship.

REPRODUCIBILITY STATEMENT

All quantitative claims in this paper can be reproduced from three components that accompany the submission. The experimental protocol (models, samplers, step budgets, guidance scale, prompts and the timing protocol) is specified in Section 4 and Appendix C; the predictor is given as an explicit closed-form expression (Eqs. 2 and 3) with its per-step behaviour in Appendix Table 5; the complete computed-step schedules are listed in Appendix Table 33 and the design rule behind them in Appendix G. Evaluation scripts, prompt lists, precomputed prompt embeddings and the calibration code are included in the supplementary material, and every experiment uses fixed seeds; the attribution claims additionally report per-image paired statistics and three-seed spreads in Appendices D and D.4. Complete proofs of all theoretical claims are in Appendix H.

REFERENCES

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations (ICLR)*, 2024.

Huanpeng Chu, Wei Wu, Guanyu Feng, and Yutao Zhang. OmniCache: A trajectory-oriented global perspective on training-free cache reuse for diffusion transformer models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16302–16312, 2025.

Bowen Cui, Yuanbin Wang, Huajiang Xu, Biaolong Chen, Aixi Zhang, Hao Jiang, Zhengzheng Jin, Xu Liu, and Pipei Huang. Denoising as path planning: Training-free acceleration of diffusion models with DPCache. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43632–43642, 2026.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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

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

Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, Suha Doshi, et al. Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation. *arXiv preprint arXiv:2402.01306*, 2024.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Haozhe Liu, Wentian Zhang, Jinheng Xie, Francesco Faccio, Mengmeng Xu, Tao Xiang, Mike Zheng Shou, Juan-Manuel Perez-Rua, and Jürgen Schmidhuber. Faster diffusion via temporal attention decomposition. *Transactions on Machine Learning Research (TMLR)*, 2025b.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863, 2025c.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Xuewen Liu, Zhikai Li, and Qingyi Gu. CacheQuant: Comprehensively accelerated diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23269–23280, 2025d.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations (ICLR)*, 2025.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

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

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

Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

Zhirong Shen, Rui Huang, Jiacheng Liu, Chang Zou, Peiliang Cai, Shikang Zheng, Zhengyi Shi, Liang Feng, and Linfeng Zhang. Beyond fixed formulas: Data-driven linear predictor for efficient diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30792–30801, 2026.

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

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

Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

Patrick von Platen, Suraj Patil, Anton Lozhkov, et al. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2024.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

Jiazheng Xu, Xiao Liu, Jianqiao Lu, et al. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pp. 4832–4839, 2019.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research (TMLR)*, 2022.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations (ICLR)*, 2025.

APPENDIX CONTENTS

Appendix section	Page
A. Extended related work	14
— Feature caching for diffusion models, in full	14
— Data-driven linear representations of nonlinear dynamic...	14
B. Datasets and evaluation metrics	15
— COCO val2017	15
— MJHQ-30K	15
— CLIP score	15
— Sharpness	15
— FID	15
C. Implementation details and timing protocol	15
— Calibration cost	15
— Prediction cost	15
— Timing	15
— Second backbone (SD3.5-Medium)	15
D. Additional results	16
— Qualitative effect	16
— Calibration	16
— Generality on this backbone: budget, resolution, solver	20
— Predictor ablation on this backbone	20
— Gating on this backbone	21
— Operator analysis on this backbone	21
— Local Lipschitz constant ρ on this backbone	22
— A third prompt set on this backbone	22
— Seed variability and calibration size on this backbone	23
— Cross-machine replication	23
— Seed variability	24
— Baseline operating points	24
— Calibration-set size	24
— PartiPrompts	25
— The design rule (G1)–(G4), in full	25
— The proxy $e(\mathcal{S})$ and its reliability	26
— Schedule search and sensitivity	26
— Scope of the proxy	26
— Residual-guided placement of new operating points	26
— Weight-variance bootstrap (Section 3.5)	27
— Operator spectrum (gap bound)	27
— Local Lipschitz constant ρ (skip-run envelope)	27
E. Alternative predictors and gates	28
— Higher-order fits	28
— Least squares versus finite differences	29
— Per-block forecasting	29
— The depth–time hybrid	29
— The clean-signal reparameterization	30
F. Learning-to-Cache under our protocol	30
— Protocol	30
— Results	30
G. Computed-step schedules	31
— Reproducing the schedules	31
H. Proofs	31
— Proof of Proposition 1 (exactness on polynomial observa...	31
— Proof of Proposition 2 (sensitivity of the forecast)	32
— Proof of Proposition 3 (variance reduction by calibration)	32
— Proof of Proposition 4 (error accumulation over a skip ...	32

A EXTENDED RELATED WORK

Feature caching for diffusion models, in full. DeepCache (Ma et al., 2024b) reuses U-Net features across steps on a hand-designed interval. For diffusion transformers, ToCa (Zou et al., 2025) chooses which tokens to cache, and TGATE (Liu et al., 2025b) caches attention outputs once they converge; TeaCache (Liu et al., 2025a) gates each step on a statistic of the timestep embedding, and advances it by a first-order estimator on the residual stream. Recomputing the first block at every step and caching the residual of the remaining blocks preserves detail by construction; that method is distributed with the Diffusers library rather than described in a publication, so we treat it as an engineering baseline and call it FBCache. Recent methods shape the temporal structure of the cache: TaylorSeer (Liu et al., 2025c) replaces reuse by Taylor extrapolation, CacheQuant (Liu et al., 2025d) jointly optimises caching and quantisation through a dynamic-programming schedule, OmniCache (Chu et al., 2025) distributes reuse over the whole trajectory, and FasterCache (Lyu et al., 2025) targets video diffusion with a frequency-aware cache over the guidance branches. Closest to us, L^2P (Shen et al., 2026) observes that forecasting caches reduce to fixed linear combinations of past features and fits per-timestep weights by gradient descent, while DPCache (Cui et al., 2026) selects the computed steps by dynamic programming over a calibration cost tensor and fills the remaining steps by Taylor prediction. We estimate the same kind of linear predictor in closed form, with no optimisation and no hyperparameters, and we hold the calibration set fixed while varying the schedule, which lets us attribute quality to each axis. L^2P reports that its gradient fit converges in roughly twenty seconds on a single GPU (Shen et al., 2026). Both fits are cheap; the difference is the form of the estimator—closed-form algebra versus gradient descent. A structural comparison of the families, with the rule each one uses to fill a skipped step, the schedule it uses, and the preparation it requires, is Table 22.

Data-driven linear representations of nonlinear dynamics, in full. The Koopman operator framework lifts a nonlinear flow into a space of observables on which the dynamics act linearly (Brunton et al., 2022). Dynamic mode decomposition estimates the resulting operator from snapshot data by least squares (Schmid, 2010; Tu et al., 2014; Williams et al., 2015), and learned-observable variants extend it to deep models (Yeung et al., 2019). Operator-learning approaches, including Fourier neural operators (Li et al., 2021) and DeepONet (Lu et al., 2021), similarly fit function-space maps from data. We borrow the *estimation principle* of this line—a global least-squares fit of a linear input–output map—and apply it to the sampling trajectory of a diffusion transformer, while deliberately *not* constructing an operator, learning observables, or performing modal decomposition, since the trajectory window is short and the observable is the full transformer output. The method is therefore a closed-form least-squares forecast, and the contribution we claim is the controlled attribution study built around it.

Table 5: Per-step behaviour of the proposed predictor (the "two lines" of Section 3). Both branches return a tensor with the same shape and semantics as the transformer output, so the surrounding solver is untouched.

Step type	Computation
$t \in \mathcal{S}$ (full)	$f_t = \epsilon_\theta(x_t, t, c)$; append (t, f_t) to $\mathcal{H}$
$t \notin \mathcal{S}$ (skip)	$\hat{f}_t = \sum_i w_i(t) f_{s_i}$ with w from Eq. 2 or Eq. 3; reuse $\hat{f}_t$ as f_t

B DATASETS AND EVALUATION METRICS

COCO val2017. We use the val2017 split of MS COCO (Lin et al., 2014) and take the first 200 captions of a fixed prompt file as our prompt set, so that the prompt distribution is identical for every method. Reference statistics for FID are computed from 5,000 val2017 images.

MJHQ-30K. MJHQ-30K contains 30,000 images in ten categories (animals, art, fashion, food, indoor, landscape, logo, people, plants, vehicles), each paired with the prompt that generated it. We sample 20 prompts per category for a total of 200, and generate one image for each.

CLIP score. Images and prompts are embedded with CLIP ViT-B/32 (Radford et al., 2021) and scored by cosine similarity; we report the mean over the 200 images. Because CLIP is trained on image–text alignment, it is largely insensitive to high-frequency degradation; this is why we report sharpness alongside it.

Sharpness. For an image I we convert to grayscale and compute the variance of the absolute Laplacian response, $\mathrm{Var}\big(|4I_{i,j} - I_{i-1,j} - I_{i+1,j} - I_{i,j-1} - I_{i,j+1}|\big)$, averaged over the images generated by a run. The statistic responds to both genuine texture and noise, so we always inspect crops qualitatively (Figure 4); we only report shifts of this statistic that are corroborated by visual inspection.

FID. FID (Heusel et al., 2017) is computed with Inception-V3 features against the COCO reference statistics. At $n = 200$ the estimator variance of FID exceeds the differences between the caching methods we compare, so we report it for completeness and base no conclusion on it.

C IMPLEMENTATION DETAILS AND TIMING PROTOCOL

All experiments use PixArt-α (Chen et al., 2024) at 512×512 with $N = 20$ DDIM (Song et al., 2021) steps and classifier-free guidance (Dhariwal & Nichol, 2021) scale 4.5, in half precision, on a single NVIDIA GeForce RTX 4080 Laptop GPU with 12 GB of memory. Prompt embeddings are precomputed once with the frozen T5 encoder, which is therefore not part of the generation loop; all methods share the same precomputed embeddings, the same sampler and the same seed.

Calibration cost. The calibration set consists of $P = 20$ prompts. Their trajectories are collected with the unmodified model, and the weights of Eq. 3 are obtained by solving an $m \times m$ normal-equation system for each history length m. There is no gradient training, no parameter update and no modification of the pretrained network; the whole calibration takes seconds on the same GPU. We calibrate once per sampler budget and reuse the same weights for every schedule, every computed-step budget and both benchmarks at that sampler budget; moving to a different sampler budget requires re-fitting on its trajectories (Section 4.3).

Prediction cost. At a skipped step the predictor evaluates a weighted sum over at most eight cached tensors of the same shape as the transformer output, on top of the solver update that any sampler performs. The measured wall-clock speedup is therefore close to, but below, N/K.

Timing. Reported speedups are per-image wall-clock times including the VAE decoding step. Repeated timings of an identical configuration on a laptop GPU can differ by up to 25%, so all methods reported in a single table are timed within one batch, and timings are not compared across batches. Absolute per-image times move with the device state; the ordering within a batch is stable.

Second backbone (SD3.5-Medium). For the cross-architecture check of Section 4.4 we load the 4-bit ($q4_k_s$) SD3.5-Medium transformer into the reference `diffusers` MMDiT implementation using the official checkpoint-to-Diffusers name mapping, and pair it with the released VAE (scaling 1.5305, shift 0.0609) and text stack (CLIP-L, CLIP-G and T5-XXL, whose 77-token CLIP context is zero-padded to 4096 and concatenated with the 256-token T5 context to a 333×4096 sequence, with a 2048-d pooled vector). We sample with a flow-matching Euler solver over a 28-step trajectory, compute the 8 steps $[0, 1, 3, 6, 10, 15, 20, 25]$ and forecast the remaining 20; the operator is the same

closed form, calibrated on 20 prompts of this model. The cached and uncached configurations differ *only* in which steps are computed, so the 4-bit quantization is common to both and does not enter the comparison. (The released `clip_l` omits the text-projection matrix; we initialize it to identity. Because both configurations share the identical text stack, this does not affect the reported caching effect.) We report CLIP and the Laplacian sharpness statistic computed exactly as for PixArt.

D ADDITIONAL RESULTS

D.1 THE ATTRIBUTION CONTROLS IN FULL

Panels (b) and (c) of the attribution control of Section 4.5, on COCO val2017 with 200 prompts.

Table 6: Predictor variants at a fixed 8-step schedule $[0, 1, 2, 4, 7, 11, 15, 18]$, *no calibration*: every row is the per-image least-squares fit of Eq. 2, so all differences are attributable to the predictor alone. The calibrated counterpart of the first row is in Table 21.

Predictor	CLIP ↑	Sharpness ↑	Outcome
Affine least squares (ours)	**30.06**	137.3	adopted
Recency-weighted least squares (γ=0.5)	30.04	265.1	no gain over affine
Predictor–corrector feedback (λ=0.7)	29.21	208.4	not adopted
Quadratic least squares	26.18	483.0	not adopted
Clean-signal reparameterization	20.46	828.9	infeasible[†]

[†]The transformer we use emits an 8-channel output rather than the 4-channel noise prediction the reparameterization assumes, so its algebraic inversion is not defined for this checkpoint.

Table 7: Gate threshold sweep on the same benchmark. The gate is the same first-order statistic FBCache uses, applied to the transformer input; it is calibrated on a small probe (which selects the threshold) and then applied per image. All gated rows are dominated by the fixed 6-step schedule at a comparable or larger compute budget.

Scheduling rule	Skip rate	CLIP ↑
Fixed $[0, 1, 2, 5, 9, 14]$	0.70	**29.94**
Adaptive gate, τ=0.04	0.59	22.27
Adaptive gate, τ=0.05	0.72	20.34
Adaptive gate, τ=0.07	0.80	19.53

Qualitative effect. Figure 4 shows pixel-level crops of the uncached model, FBCache, and our 7/8-step points: the blur all caching methods introduce is visible as missing texture at 7 steps and largely recovered at 8, tracking the sharpness column of main-text Table 1.

D.2 HUMAN-PREFERENCE VALIDATION ON PIXART-α

Calibration. main-text Figure 2 isolates one-shot calibration by fitting the same linear model with and without the cross-prompt pooling of Eq. 3, at three step budgets. Alignment is essentially unchanged, whereas sharpness improves by 2.0×, 2.9× and 3.5× at 6, 7 and 8 steps. The cross-trajectory estimate of the same operator therefore produces visibly more detailed images without moving the alignment score. The variance-reduction statement of Section 3.5 bounds the variance of the fitted *weights*; the sharpness change itself we report as an empirical, reproducible observation (a single scalar sharpness statistic is itself noisy at this sample size, Appendix B) that alignment-only evaluations would not detect, and a mechanistic account of it is left to future work. On SD3.5, by contrast, the calibration saturates immediately: P=1 already matches P=20 to within 0.09 CLIP (Table 19), so the operator itself, not its estimation, is what this backbone constrains.

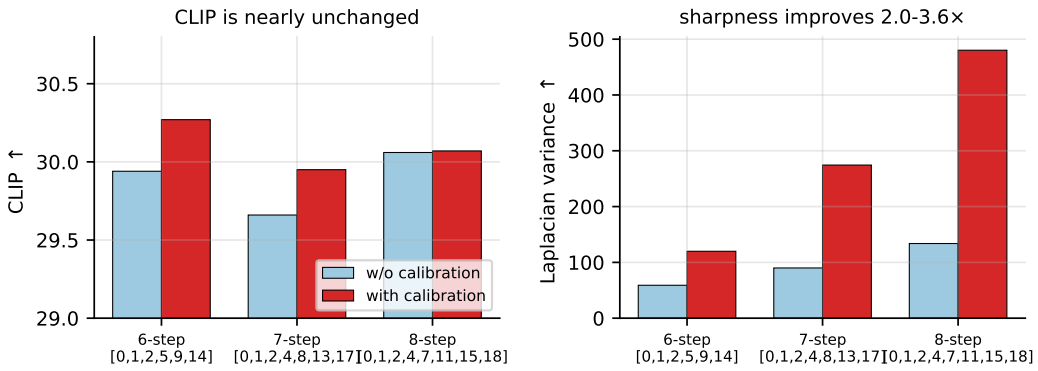


Figure 2: One-shot calibration, COCO val2017 (PixArt-α). Left: alignment is nearly unchanged at every step budget. Right: sharpness improves by 2.0–3.5$\times$ at the same step budgets. Both panels use identical schedules and identical numbers of computed steps.

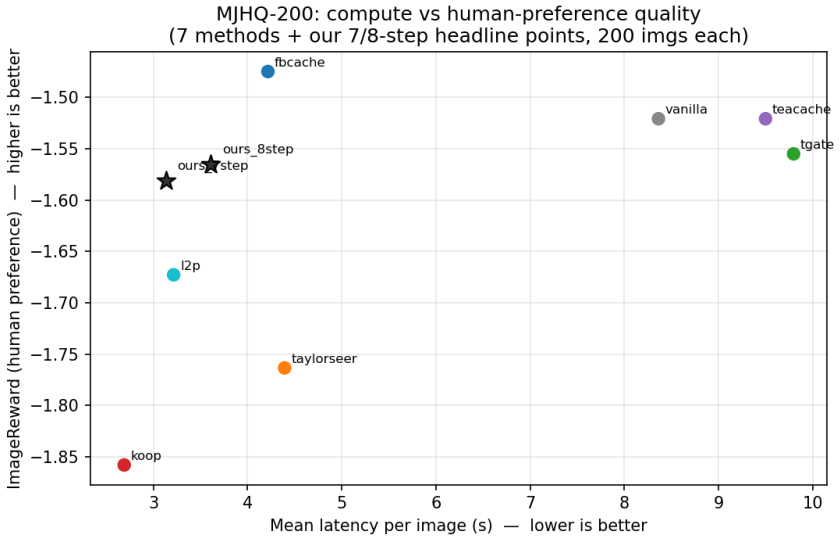


Figure 3: Compute vs human-preference quality on MJHQ-200 (PixArt-α). Stars are our 7/8-step headline operating points; they occupy the far-left (fastest) end at a human-preference level statistically indistinguishable from the uncached model and FBCache.

D.3 Generality probes on PixArt-α

The three probes of Section 4.3 are tabulated here for PixArt-α (Table 8).

D.4 Second backbone: full benchmark on SD3.5-Medium

Section 4.4 reports the cross-architecture check on the two configurations that matter for the claim. Here we give the *complete* replication of the main protocol on that backbone: six of the seven baselines of main-text Table 1 ported to the MMDiT-X block and run on the same 200 MJHQ prompts, at the same resolution, with the same paired seeds, plus the human-preference metric.

Three protocol details of Table 11 differ from main-text Table 1. *(i)* The trajectory is 28 flow-matching steps rather than 20 DDIM steps, so "steps" and speedups are not comparable across the two tables; Tables 10 and 11 report results within this backbone. *(ii)* Two families of accelerators appear. Ours, L^{2}P and DPCache skip whole transformer evaluations (so the s/img gain tracks the

Table 8: Generality probes on COCO val2017 (PixArt-α). "Budget 30" uses 30 DDIM steps: the weights calibrated at 20 steps transfer badly (alignment -1.7 against the recalibrated point, and the Laplacian statistic rises to 2274, i.e. high-frequency artefacts), but re-fitting the same closed-form operator on 20 prompts of the new budget restores them. "768" is 100 prompts at 768×768; FID is omitted there because no matched reference statistics exist at that resolution. "DPM" is DPM-Solver++ with 10 steps, with and without our cache on top.

Setting	Method	s/img	Speedup ↑	CLIP ↑
Budget 30	Uncached	12.372	1.00×	29.16
Budget 30	FBCache	4.961	2.49×	**30.29**
Budget 30	Ours, weights from 20 steps	3.912	3.16×	28.35
Budget 30	Ours, recalibrated	3.870	**3.20**×	30.06
768	Uncached	8.310	1.00×	29.91
768	FBCache	4.213	1.97×	**30.40**
768	Ours (7 steps)	3.160	**2.63**×	30.12
DPM 10	Uncached	4.358	1.89×	**30.88**
DPM 10	Ours on top	2.761	**2.99**×	30.67



Figure 4: Qualitative comparison on COCO val2017 ("a bowl filled with many shiny green apples"), centre crop at 2×. From left: uncached, FBCache (1.91×), our 7-step configuration (2.57×), our 8-step configuration (2.29×). Specular highlights and the folds of the plastic bag are preserved by FBCache and by our 8-step configuration, and visibly softened at 7 steps—the trade-off quantified in main-text Table 1.

computed-step count), whereas FBCache, TeaCache, TGATE and TaylorSeer still call the model at every step and only skip *sub-module* compute inside the blocks. Their reported speedup is therefore wall-clock only and cannot be read off the computed-step column. *(iii)* Every threshold- or interval-based baseline is *re-calibrated* to its best speed/quality trade-off on this backbone.

On COCO the eight-step point loses 0.07 CLIP while being 1.8× faster than FBCache, which loses ten times as much alignment (-0.72) and retains 51% of the reference sharpness. Seven steps is statistically indistinguishable from eight here (-0.06), and even six steps (4.48×) stays within 0.62 CLIP; L^{2}P is again the only baseline that matches us on alignment, at 2.38× and with the same preparation cost it had on PixArt. TGATE yields no speedup on this backbone (0.95×) and is marginally slower than the uncached model, because it cannot skip the fused attention and still incurs the caching overhead.

The ordering of the baselines reproduces main-text Table 1 closely. TGATE again yields no speedup (0.95× here, 1.04× on PixArt). FBCache is again the closest competitor on speed, and again trades a large amount of detail for it: at 1.88× it retains only 60% of the reference sharpness, whereas our 8-step point is 1.8× faster than FBCache and keeps 97%. L^{2}P is the one baseline that beats us on the quality axis—+0.18 CLIP at 2.42×—and it is exactly the baseline that obtains its quality with the most preparation (gradient-descent training of the forecast weights plus calibration). On the speed–quality frontier, no baseline matches our operating point: at 3.39× there is no competitor in the replicated table. This is the same conclusion as on PixArt, with a larger speedup: the same

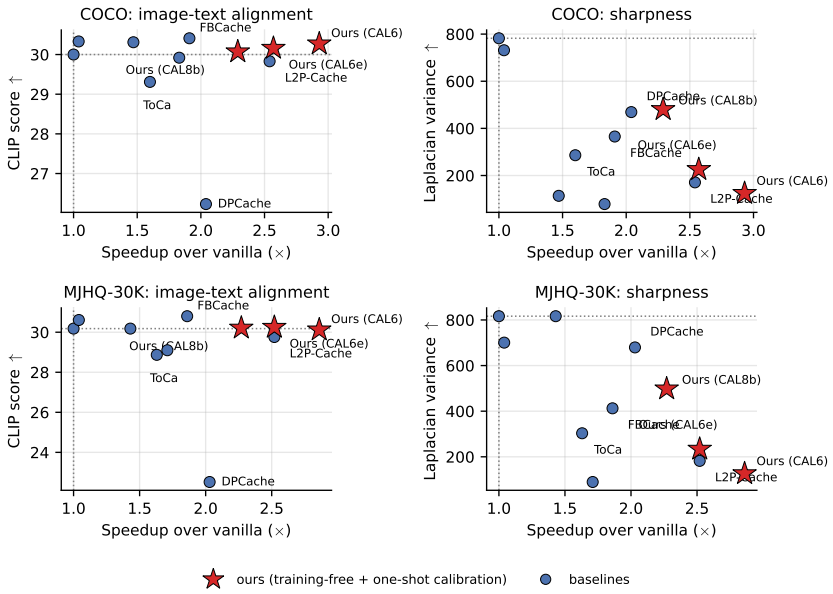


Figure 5: Speed–quality trade-offs on both benchmarks. Top row: COCO val2017; bottom row: MJHQ-30K. Left column: image–text alignment; right column: sharpness. Stars are the proposed method at three step budgets (6, 7, 8 computed steps); dotted lines mark the uncached model. The proposed method occupies the rightmost part of each panel, i.e. it is the fastest at every quality level it attains.

Table 9: (a) Adaptive gating versus a fixed schedule on SD3.5-Medium, COCO val2017, 200 prompts, one timing batch. Every fixed schedule at a comparable or larger speedup beats every gate — the PixArt conclusion of main-text Table 4 reproduced on a different architecture and a different sampler family. (b) Generality on the same backbone: each row group is its own timing batch, so speedups are comparable within a group. At a 30-step budget and at 768px the method's CLIP score exceeds the uncached model's.

(a) Step selection, SD3.5					(b) Generality, SD3.5			
Setting	*s*/img	Speedup	CLIP	Sharp.	Setting	*s*/img	Speedup	CLIP
Fixed sched., 8 steps	1.10	3.39×	31.53	785.3	Budget 30, uncached	4.08	1.00×	31.62
Fixed sched., 7 steps	0.98	**3.83**×	**31.53**	908.2	Budget 30, ours 9/30	1.26	3.24×	**31.71**
Gate τ=0.02	3.98	0.93×	31.59	819.8	768px, uncached	7.74	1.00×	31.39
Gate τ=0.05	2.78	1.33×	31.28	896.5	768px, ours 8/28	2.40	3.23×	**31.52**
Gate τ=0.10	0.98	3.78×	31.08	1000.1	768px, ours 7/28	2.10	**3.68**×	31.35

eight-step budget that yields 2.32–2.67× on the 20-step PixArt solver yields 3.39× here, i.e. the flow-matching trajectory carries more redundancy.

The optimal budget is data dependent: on MJHQ, seven and six steps cost 0.86 and 1.42 CLIP, whereas on COCO seven steps is free (−0.06). Consequently, a budget calibrated on one prompt distribution may not transfer to another, consistent with the operator being fitted to a trajectory rather than to a model.

The human-preference metric agrees with CLIP: across the ten rows the two metrics have Spearman $\rho = 0.94$ ($p < 10^{-3}$), tighter than the $\rho = 0.89$ we measure on PixArt. Our operating point is third only to the uncached reference and L^2P, ahead of four of the six baselines, while being 1.4× faster than L^2P and 1.8× faster than FBCache. The tighter budgets are penalised more by the preference metric than by CLIP: the 7-step point sits between FBCache and TeaCache, and the 6-

Table 10: COCO val2017 replication on SD3.5-Medium (MMDiT-X), 200 prompts, 512px, CFG 4.5, 28-step flow-matching trajectory, paired seeds. Every latency in this table (and in every other table below) comes from one single back-to-back batch of all rows, not from the per-run logs, so the speedup column is internally comparable. The four cache-based baselines still call the transformer at every step and skip only sub-module compute, which is why their speedup cannot be read off a computed-step count.

Method	Computed	s/img	Speedup ↑	CLIP ↑	ΔCLIP	Sharpness ↑
Uncached	28/28	3.74	1.00×	**31.59**	–	819.8
Ours (8 steps)	8/28	1.10	3.39×	31.53	−0.07	785.3
Ours (7 steps)	7/28	**0.98**	3.83×	31.53	−0.06	908.2
Ours (6 steps)	6/28	**0.84**	**4.48**×	30.98	−0.62	972.2
TeaCache	28/28	1.98	1.89×	31.21	−0.38	899.1
FBCache	28/28	1.97	1.89×	30.87	−0.72	414.0
TaylorSeer	28/28	2.63	1.42×	31.38	−0.21	997.7
L^2P	11/28	1.57	2.38×	31.58	−0.01	782.4
DPCache	13/28	1.86	2.01×	29.86	−1.73	964.9
TGATE	28/28	3.98	0.94×	30.98	−0.61	653.8

Table 11: Full replication of main-text Table 1 on SD3.5-Medium (MMDiT-X), 512px, the 200 MJHQ-30K prompts, CFG 4.5, 28-step flow-matching trajectory, paired seeds. Latencies are wall-clock per image from one single back-to-back batch containing every row of this table. Ours computes 8/7/6 of the 28 steps and forecasts the rest with the same one-shot closed-form operator. All rows run the identical 4-bit weights, so the CLIP difference isolates the caching effect.

Method	Computed	s/img	Speedup ↑	CLIP ↑	ΔCLIP	Sharpness ↑	Extra prep.
Uncached	28/28	3.78	1.00×	**32.15**	–	818.9	–
Ours (8 steps)	8/28	1.12	3.39×	31.81	−0.34	795.4	one-shot
Ours (7 steps)	7/28	0.98	3.88×	31.29	−0.86	922.5	one-shot
Ours (6 steps)	6/28	**0.83**	**4.53**×	30.73	−1.42	1107.0	one-shot
L^2P	11/28	1.56	2.42×	31.99	−0.16	794.8	calib.+training
DPCache	13/28	2.03	1.86×	30.82	−1.33	953.1	calibration
TeaCache	28/28	1.99	1.90×	31.36	−0.79	732.3	none
FBCache	28/28	2.01	1.88×	31.09	−1.06	491.7	none
TaylorSeer	28/28	2.62	1.45×	31.74	−0.41	901.5	none
TGATE	28/28	3.99	0.95×	30.61	−1.54	728.8	none

step point between TeaCache and DPCache. The tighter budgets are therefore penalised more by the preference metric than by CLIP: this indicates that while CLIP reports these images as well aligned, the human-preference metric captures a perceptual degradation at higher speedups.

Generality on this backbone: budget, resolution, solver. The three probes of Section 4.3 are repeated here on SD3.5-Medium. Each row group is timed in its own single batch, so speedups are comparable within a group and not across groups; each new budget or resolution re-fits the same closed-form operator on 20 prompts of that setting.

Two of the three transfer. At the 30-step budget, re-fitting the operator on 20 prompts of the new budget produces a point that is *both* the fastest and the best aligned: 3.24× with CLIP 31.71 against the uncached 31.62. At 768px the method keeps 3.68× (7 steps) or 3.23× (8 steps) with alignment within noise of the uncached model, and its position relative to FBCache is unchanged—FBCache again collapses to 42% of the reference sharpness. The solver axis behaves as expected: a second-order update at the same eight evaluations changes CLIP by +0.03 (paired t-test p=0.70), i.e. at this budget the integration order is not the binding constraint, and the update composes with skipping without disturbing the speedup (3.38× vs. 3.37×).

Predictor ablation on this backbone. Table 14 holds the 8-computed-step schedule fixed and changes only what fills a skipped step. On SD3.5 the predictor is almost irrelevant: verbatim reuse,

Table 12: ImageReward (higher is better) on the same SD3.5-Medium MJHQ-200 runs as Table 11, mean±std over 200 images. As on PixArt, the metric is reported for internal comparison only: its absolute scale depends on the backbone.

Method	CLIP ↑	ImageReward ↑
Uncached	**32.15**	$\mathbf{+0.571 \pm 0.952}$
L^2P	31.99	$+0.432 \pm 0.988$
Ours (8 steps)	31.81	$+0.350 \pm 0.976$
TaylorSeer	31.74	$+0.290 \pm 0.978$
FBCache	31.09	$+0.197 \pm 0.968$
Ours (7 steps)	31.29	$+0.132 \pm 1.017$
TeaCache	31.36	$+0.120 \pm 1.032$
Ours (6 steps)	30.73	-0.120 ± 1.039
DPCache	30.82	-0.133 ± 0.979
TGATE	30.61	-0.189 ± 1.046

Table 13: Generality probes on SD3.5-Medium (MMDiT-X), COCO val2017, CFG 4.5. "Budget 30" is a 30-step flow-matching trajectory; "768" is 100 prompts at 768×768; "AB2" swaps the first-order flow-matching Euler update for a second-order Adams–Bashforth update at the *same* number of transformer evaluations, so it isolates the solver axis from the compute axis.

Setting	Method	Computed	s/img	Speedup ↑	CLIP ↑
Budget 30	Uncached	30/30	4.08	1.00×	31.62
	FBCache	30/30	2.01	2.03×	30.91
	Ours, recalibrated	9/30	1.26	**3.24**×	**31.71**
768	Uncached	28/28	7.74	1.00×	31.39
	FBCache	28/28	4.12	1.88×	30.97
	Ours (8 steps)	8/28	2.40	3.23×	**31.52**
	Ours (7 steps)	7/28	2.10	**3.68**×	31.35
AB2 (2nd order)	Uncached (Euler)	28/28	3.70	1.00×	31.59
	Ours (Euler)	8/28	1.10	3.37×	31.53
	Ours (AB2 update)	8/28	1.10	3.38×	31.56

recency weighting, the predictor–corrector blend and the affine least-squares fit are all within 0.10 CLIP of each other and none of the differences is significant (paired t-tests, n=200). Only the quadratic fit is clearly harmful—−2.11 CLIP—and it does so while inflating the Laplacian statistic to 1117, the same failure mode we documented on PixArt. All rows cost essentially the same (1.09–1.12 s), so the comparison is schedule-controlled.

This cross-backbone difference is consistent with the attribution claim of Section 4.5: on SD3.5 the predictor matters less than on PixArt. On PixArt the forecaster was worth roughly a factor of two in sharpness at fixed alignment; on the smoother flow-matching trajectory of SD3.5 the predictor contributes little at this budget, so the quality difference between our 8-step point and FBCache is governed by *which steps are computed*.

Gating on this backbone. Replacing the fixed schedule with a per-image adaptive gate—the first-order statistic FBCache uses, applied to the transformer input, with the skip run capped at 5 steps—is again dominated (Appendix Table 9a). At matched cost, the fixed 7-step schedule and the gate at τ=0.10 both run at 0.98 s/image, and the schedule loses 0.06 CLIP while the gate loses 0.45. At τ=0.02 the gate is *slower than the uncached model* (0.93×) because it elects to compute almost every step. A fixed schedule outperforms per-image gating at matched cost.

Operator analysis on this backbone. The two theoretical diagnostics of Appendix D.5 and D.5 were recomputed on SD3.5 calibration trajectories. The bootstrap weight standard deviation falls from 0.0212 at P=5 to 0.0183 at P=10 and 0.0091 at P=20, a 2.33× reduction, consistent with the predicted $P^{-1/2}$ decay. The spectral radius of the companion matrix induced

Table 14: Predictor ablation on SD3.5-Medium, COCO val2017, 200 prompts, with the *one fixed* 8*-computed-step schedule* $[0, 1, 3, 6, 10, 15, 20, 25]$ for every row. ΔCLIP is the paired mean difference against the affine least-squares row, with the paired t-test p-value in brackets. Latencies from a single back-to-back batch.

Skipped step uses	s/img	Speedup ↑	CLIP ↑	ΔCLIP	Sharpn. ↑
Affine least squares (ours)	1.10	3.37×	31.53	–	785.3
Verbatim reuse (no forecast)	1.09	3.41×	31.61	+0.08 [0.29]	845.8
Affine LS with AB2 solver	1.10	3.38×	31.56	+0.03 [0.70]	753.5
Recency-weighted LS (γ=0.5)	1.11	3.33×	31.62	+0.10 [0.25]	876.2
Predictor–corrector (λ=0.7)	1.12	3.31×	31.60	+0.07 [0.24]	827.7
Quadratic least squares	1.12	3.32×	29.41	−2.11 [$<10^{-4}$]	1117.3

by the fitted weights stays within 10% of unity for every forecast window we fit, L=2, . . . , 8 $(0.987, 0.996, 0.993, 0.988, 0.978, 0.905, 1.001)$—numerically neutral throughout. In contrast, it does *not* rise at L=8 the way it does on PixArt (1.367 there), so on this backbone the operator is stable even at the largest window we use, and the gap bound of Eq. 4 is conservative rather than necessary.

Table 15: Operator diagnostics on SD3.5-Medium calibration trajectories (COCO-20). Left: bootstrap standard deviation of the fitted weights versus calibration size P (with-replacement, 20 replicates); the predicted $P^{-1/2}$ decay is reproduced. Right: spectral radius of the companion matrix induced by the fitted weights versus forecast-window length L; neutral (within 10% of 1) for every L we fit, unlike PixArt where L=8 rises to 1.367.

P	Wt. std ↓	vs. P=5	L	Spectr. radius	Regime
5	0.0212	1.00×	2	0.987	neutral
10	0.0183	1.16×	5	0.988	neutral
20	0.0091	**2.33**×	8	1.001	neutral

Local Lipschitz constant ρ on this backbone. The accumulation bound of Section 3.5 was measured on SD3.5 exactly as in Appendix D.5: for 12 prompts we perturb a single block's image-stream output at a denoising step $t \in \{8, 12, 16\}$ by 1% of its Frobenius norm and measure the magnification of the *same* block's output at t+1, for blocks $\{0, 5, 11, 17, 23\}$—180 measurements.

Table 16: Measured per-step magnification ρ of a perturbed block output on SD3.5-Medium (180 measurements: 12 prompts × 5 blocks × 3 steps, 512px). On this backbone *every* perturbation is contractive: the largest single-step magnification over all 180 samples is 0.41.

	median	75th pct.	max	fraction < 1
SD3.5-Medium (this work)	0.177	0.221	0.405	100%
PixArt-α (Appendix D.5)	0.45	–	3.6	75%

The result is markedly stronger than on PixArt. There, a quarter of the perturbations *amplified* the deviation, with a worst case of 3.6, so the argument for capping the gap rested on the bound staying finite. Here all 180 samples contract, and the worst case is 0.405: a skip run of length k multiplies the forecast error by at most $0.405^k < 1$—the error is not merely bounded, it decays geometrically with the length of the run. This is the quantitative counterpart of the neutral spectral radius observed for every window L above, and it explains why the flow-matching trajectory tolerates the same eight-step budget with a smaller quality cost than the DDIM trajectory did.

A third prompt set on this backbone. PartiPrompts is the third prompt distribution used on PixArt; Table 17 repeats it here. The eight-step point loses 0.01 CLIP and keeps 99% of the reference sharpness, while FBCache loses 0.29 and keeps 61%. PartiPrompts is the hardest of the three

sets for every method (the uncached CLIP is 28.7 against 31.6 on COCO and 32.2 on MJHQ), and it is where the gap between our point and FBCache is smallest in relative terms—but the ordering is unchanged.

Table 17: PartiPrompts, 200 prompts, SD3.5-Medium, 512px, 28-step trajectory, paired seeds. The images were generated on the second machine; CLIP and sharpness are device-independent (we verify agreement to within 0.27 / 4% in the replication below), but the latency column was re-measured on the primary machine in one back-to-back batch using *medians* over 30 images, because a single image in the 8-step row was a 2× outlier (mean 1.22 s against median 1.07 s).

Method	s/img	Speedup ↑	CLIP ↑	ΔCLIP	Sharpness ↑
Uncached	3.68	1.00×	**28.69**	–	1060.0
Ours (8 steps)	**1.07**	**3.44**×	28.68	−0.01	1050.7
L^2P	1.54	2.39×	28.67	−0.02	1042.7
FBCache	1.80	2.05×	28.40	−0.29	650.4
DPCache	1.81	2.04×	27.99	−0.70	1253.7
TaylorSeer	2.57	1.43×	28.53	−0.16	1280.4
TGATE	3.92	0.94×	28.80	+0.11	1048.3

Seed variability and calibration size on this backbone. Table 18 re-runs the three operating points at two further seeds. The cross-seed standard deviation is 0.10–0.13 CLIP, so the seed-0 gaps of Table 11 (−0.34, −0.86, −1.42) are 3–10× the seed noise: the ordering of the budgets is not an artefact of one seed. Table 19 varies the number P of calibration prompts. Alignment is flat to within 0.09 CLIP across P=1, . . . , 20, i.e. on this backbone a *single* calibration trajectory already reproduces the full P=20 operator—one-shot calibration can be made essentially free here, which is a stronger statement than the $P^{-1/2}$ weight-stability decay of Table 15 would suggest.

Table 18: Seed variability on SD3.5-Medium, MJHQ-200, mean±std over three seeds $(0, 100, 200)$. ΔCLIP is against the seed-0 uncached model (CLIP 32.15). The cross-seed spread is 0.10–0.13 CLIP, well below every gap that matters.

Configuration	CLIP ↑	ΔCLIP	Sharpness ↑	per-seed CLIP
Ours (8 steps)	31.70 ± 0.11	-0.45 ± 0.11	805 ± 17	31.81, 31.59, 31.70
Ours (7 steps)	31.29 ± 0.13	-0.86 ± 0.13	961 ± 41	31.29, 31.42, 31.15
Ours (6 steps)	30.77 ± 0.10	-1.38 ± 0.10	1133 ± 35	30.73, 30.89, 30.70

Table 19: Calibration-set size P on SD3.5-Medium, MJHQ-200, fixed 8-computed-step schedule. P=20 is the headline configuration; P=1 refits the same closed form on a single trajectory. Alignment is flat in P: one prompt suffices on this backbone.

P	CLIP ↑	ΔCLIP	Sharpness ↑
1	31.83	−0.32	803.8
5	31.74	−0.41	798.4
10	31.78	−0.37	794.7
20	31.81	−0.34	795.4

Cross-machine replication. The four cache-based baselines were re-run end-to-end on a second machine (a different GPU model, `torch` 2.6 vs. 2.1.2, CUDA 12.4 vs. 12.6) on the same prompts and seeds. CLIP agrees to within 0.27 and sharpness to within 4% across machines, so none of the conclusions above depends on a single device or software stack. Timings, in contrast, are *not* comparable across machines: the second GPU thermal-throttles to 87 °C and runs 1.8× slower, which would spuriously reverse the ranking if the two sets of latencies were mixed. All numbers in Table 11 come from one machine in one batch, following the protocol of Appendix C.

Table 20: Threshold-based baselines at several operating points on COCO val2017 (200 prompts). The threshold is the only knob such methods expose; the row used in main-text Table 1 is marked.

Baseline / threshold	Speedup ↑	CLIP ↑	Sharpness ↑
TeaCache, default (used in main-text Table 1)	1.47×	30.31	113.8
TGATE, default (used in main-text Table 1)	1.04×	30.33	731.5
TaylorSeer, interval 4 (used in main-text Table 1)	1.83×	29.92	78.8

Table 21: The proposed method at every step budget we evaluated, COCO val2017. "w/o cal." is the per-image least-squares fit of Eq. 2 and "w/ cal." the cross-prompt fit of Eq. 3. *Quality* is comparable across the two columns because the images and prompts are identical; the two *speedup* columns come from different measurement batches and must not be compared (Appendix C). No 7-step $[0, 1, 2, 5, 9, 14, 18]$ row is listed without calibration because that schedule was only measured with the calibrated weights; the 2.9× sharpness gain quoted for seven steps in the main text is measured on the $[0, 1, 2, 4, 8, 13, 17]$ schedule.

		w/o calibration		with calibration		
Schedule	Speedup	CLIP	Sharp.	CLIP	Sharp.	Speedup
6-step $[0, 1, 2, 5, 9, 14]$	3.08×	29.94	63.8	30.27	124.4	2.48×
7-step $[0, 1, 2, 5, 9, 14, 18]$	–	–	–	30.16	225.8	2.64×
7-step $[0, 1, 2, 4, 8, 13, 17]$	2.35×	29.66	95.7	29.95	278.4	2.16×
8-step $[0, 1, 2, 4, 7, 11, 15, 18]$	2.42×	30.06	137.3	30.07	480.2	2.31×
9-step $[0, 1, 2, 4, 6, 9, 12, 15, 18]$	1.68×	30.24	164.9	–	–	–

Seed variability. Table 23 reports the full 200-prompt run repeated under three random seeds for the unmodified model, FBCache and our method. Per-method standard deviations are 0.05–0.12 CLIP across the three seeds (Table 23), so a gap of about 0.25 on COCO is within the seed-to-seed spread (roughly two standard deviations) and is *not* a significant difference at the resolution our seeds afford; a 0.55 gap on MJHQ-30K is several standard deviations. We call such gaps 'within noise' and treat them as indistinguishable rather than as wins or losses; this is a paired-seed threshold, not an arbitrary cut-off.

Baseline operating points. Threshold- and interval-based baselines expose one knob; sweeping it shows how sensitive they are to the operating point (Table 24). TaylorSeer's quality collapses at several adjacent intervals, and TeaCache's threshold must be pushed to about 0.01 or below before it skips appreciably on PixArt; at its fastest point its alignment drops 2 points below ours at a higher wall-clock cost.

Calibration-set size. Table 25 varies the number P of calibration prompts. Alignment improves monotonically with P (29.7 at P=5, 29.7 at P=10, 30.2 at P=20): as the variance-reduction statement predicts, more calibration reduces the variance of the fitted weights and the trajectory then tracks the truth more closely. Sharpness, however, does *not* track P monotonically—it peaks at P=5–10 (431–436) and falls to 226 at P=20. This asymmetry is the point: weight variance falls monotonically by construction (Table 29, a 1.70× reduction from P=5 to P=20), while sharpness does not, so the sharpness effect of calibration *cannot* be attributed to variance reduction. It is set instead by the schedule (Observation 3): the residual the forecast re-injects is a property of the skip geometry, and once $P \geq 5$ the weights are already clean enough that further pooling only pulls them toward the population-mean operator—trading a little re-injected high-frequency detail (436 → 226) for alignment (29.7 → 30.2), i.e. the standard bias–variance move. The schedule dependence is directly visible and of the same size: the *same* P=20 calibration reaches sharpness 278.4 under the 7-step schedule of main-text Figure 2 but 226 under the 7-step schedule used here—a 20% swing produced by the schedule alone, with the calibration held fixed. The single scalar sharpness statistic is also heavy-tailed and its per-seed standard deviation is a non-negligible fraction of its mean (Table 23), so the P=20 point is a noisy estimate; we use P=20 for the main tables because

Table 22: Relation to the compared accelerators. "One-shot" calibration fits a small linear model in closed form from ~ 20 prompts with no gradient steps and no hyperparameters. L^2P's gradient fit is comparably cheap (~ 20 s), so the difference is the estimator, not its cost.

Method	Fill-in for skipped steps	Schedule	Training	Extra prep.
DeepCache (Ma et al., 2024b)	cached features	interval	–	none
FBCache	cached residual + fresh first block	adaptive gate	–	none
TGATE (Liu et al., 2025b)	cached attention outputs	gate at convergence	–	none
TeaCache (Liu et al., 2025a)	cached features	gate on time embedding	–	none
TaylorSeer (Liu et al., 2025c)	Taylor extrapolation	interval	–	none
DPCache (Cui et al., 2026)	Taylor prediction	DP-optimal	–	calibration pass
L^2P (Shen et al., 2026)	learned linear weights	interval	gradient fit	calibration pass
Ours	**least-squares operator**	**fixed, gap-bounded**	–	**one-shot, closed form**

Table 23: Seed variability on COCO val2017 (200 prompts, three seeds). "Sharp." is the Laplacian sharpness of Appendix B.

Method	Seed	s/img	CLIP ↑	FID ↓	Sharp. ↑
Unmodified	0	8.243	30.00	154.50	782
Unmodified	1	8.322	29.93	153.28	793
Unmodified	2	8.502	30.16	152.90	822
FBCache	0	3.997	30.41	157.17	365
FBCache	1	4.353	30.27	159.51	393
FBCache	2	4.465	30.26	154.97	403
Ours (7 steps)	0	3.126	30.16	162.67	226
Ours (7 steps)	1	3.080	30.18	163.41	244
Ours (7 steps)	2	3.160	30.27	158.72	261

it maximizes alignment, and the sharpness side is governed by the schedule (Observation 3) rather than by P.

PartiPrompts. Table 26 repeats the comparison on a third benchmark, PartiPrompts (Yu et al., 2022) (200 prompts; FID is not reported because the benchmark has no reference image set). Ours attains the highest alignment while remaining the fastest, and the sharpness ordering matches the other benchmarks.

D.5 Closed-form schedule search, residual-guided selection, and operator analysis

The closed-form proxy lets us *search* the schedule space without rendering a single image. We report three zero- or low-cost analyses it enables: a schedule search and its sensitivity, a residual-guided placement of new operating points, and two empirical checks of the operator theory (a weight-variance bootstrap and the spectrum of the induced recursion).

The design rule (G1)–(G4), in full. The rule that produced every schedule used in this paper can be stated completely: (G1) compute the first two steps, since no forecast is definable before two outputs exist (our PixArt schedules anchor three, $\{0, 1, 2\}$, for extra margin); (G2) never let the gap between consecutive computed steps exceed $g_{\max} = 5$, which keeps the induced recursion in the neutral spectral regime of Table 30 (on SD3.5 the same cap is conservative: the radius stays neutral up to length 8, Table 15); (G3) spread the remaining $K - 3$ steps to cover the trajectory rather than clustering them; and (G4) rank the feasible schedules by the closed-form proxy $e(\mathcal{S})$ defined immediately below. Table 33 verifies (G1)–(G3) for every schedule used in the paper.

Table 24: Baseline sweeps on COCO val2017 (200 prompts, seed 0). All rows use the same order-0 fill as the corresponding main-table rows; "Conf." is the swept parameter.

Baseline	Conf.	s/img	CLIP ↑	FID ↓
TaylorSeer	interval 3	4.943	24.27	197.00
TaylorSeer	interval 4	4.577	29.92	161.47
TaylorSeer	interval 5	4.296	18.39	310.16
TaylorSeer	interval 6	4.146	22.35	224.91
TeaCache	threshold 0.010	5.376	30.31	161.93
TeaCache	threshold 0.005	4.906	29.84	174.73
TeaCache	threshold 0.002	4.167	28.28	203.77
TeaCache	threshold 0.150	8.509	30.00	154.50

Table 25: Calibration-set size P on COCO val2017 (our method, 7-step schedule).

P	s/img	CLIP ↑	Sharp. ↑
5	3.11	29.72	431
10	3.12	29.71	436
20	3.13	30.16	226

The proxy $e(\mathcal{S})$ and its reliability. The calibration trajectories already indicate how demanding a candidate schedule is: for a candidate $\mathcal{S}$ we forecast every skipped step from its own history with the calibrated weights and average the relative residual $e(\mathcal{S}) = \mathrm{mean}_{t\notin\mathcal{S}}\|\hat{f}_t - f_t\|/\|f_t\|$ over the calibration prompts, which requires no additional generation. Within the family of well-spread, gap-bounded schedules that (G1)–(G3) generate, e orders sharpness almost monotonically (Spearman -0.89, Kendall $\tau_b = -0.73$ across our six PixArt operating points) while being nearly uninformative about alignment (Spearman 0.49). This ordering is *local*: it does not extend to schedules that cluster their computed steps, where step geometry rather than residual magnitude governs sharpness and the correlation vanishes. We therefore use e as a cheap screen *inside* the feasible family that (G1)–(G3) generate; every operating point in our main tables is confirmed by a direct render.

Schedule search and sensitivity. For each $K \in \{6, 7, 8\}$ we sample 300 gap-bounded schedules at random and score each by the proxy residual $e(\mathcal{S})$; no images are generated. Table 27 summarises the distribution. The best proxy decreases with K (more computed steps make the forecast easier), but the *spread* grows sharply at $K{=}8$ (std 0.155, worst 0.914 for a clustered schedule), which is exactly the regime the spread condition (G3) is meant to exclude.

Scope of the proxy. The proxy is informative within the gap-bounded, well-spread family that (G1)–(G3) generate: within that family it orders sharpness while being nearly uninformative about alignment. Outside it—in particular when computed steps cluster beyond the mandatory $\{0, 1, 2\}$ anchor, violating the spread condition (G3)—sharpness becomes heavy-tailed and is governed by cluster geometry rather than by residual magnitude, so the residual-based score has nothing stable to rank (Figure 6). The spread condition is therefore a precondition for using $e(\mathcal{S})$ in (G4), and every operating point in our main tables is confirmed by a direct render regardless of its proxy value.

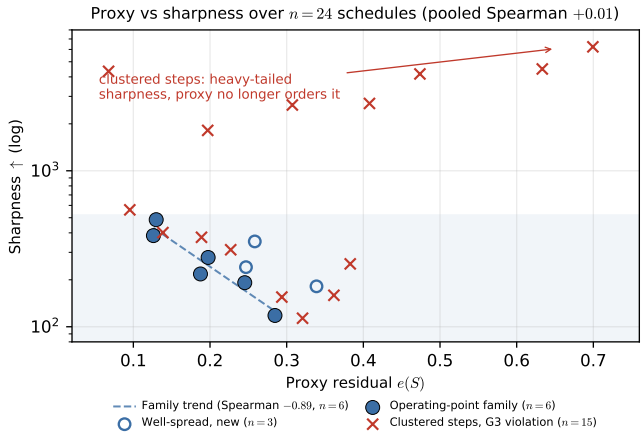
Our chosen schedules carry proxy 0.2847 ($K{=}6$), 0.1876 ($K{=}7$) and 0.1298 ($K{=}8$); the $K{=}6$ value is the least sharp of its band because we prioritised an even spread over a minimal proxy, and it is still well inside the neutral-spectral regime of Table 30.

Residual-guided placement of new operating points. The proxy is used to place a schedule not previously measured. A $K{=}6$ search returns $[0, 1, 3, 7, 12, 17]$ with proxy 0.2453—between the $K{=}6$ point (0.2847) and the $K{=}7$ point (0.1876)—predicting a sharpness between them. A single render (Table 28) gives sharpness 192.2 on COCO and 190.6 on MJHQ-30K, i.e. between the measured $K{=}6$ (124.4 / 126.3) and $K{=}7$ (225.8 / 233.8) points on both benchmarks. The same proxy places a $K{=}8$ schedule $[0, 1, 2, 4, 8, 13, 16, 18]$ with the band-best proxy 0.1262; rendered, it

Table 26: PartiPrompts (200 prompts).

Method	s/img	CLIP ↑	Sharp. ↑
Unmodified	8.397	27.17	745
TaylorSeer	4.561	27.38	67
FBCache	4.282	27.35	381
Ours (7 steps)	3.209	**27.45**	229

Figure 6: Proxy residual $e(\mathcal{S})$ against sharpness over all n=24 schedules (log scale). Filled circles are the six operating points, open circles the three well-spread additions, and crosses the 15 schedules that cluster computed steps beyond the anchor. Within the well-spread family the proxy orders sharpness (dashed trend, Spearman -0.89, n=6); once computed steps cluster—a violation of (G3)—sharpness becomes heavy-tailed and the pooled correlation collapses to $+0.01$. The proxy is a local indicator on the feasible family, not a global quality metric.



reaches sharpness 382.7 and CLIP 29.77, which is *below* the K=8 operating point (480.2 / 30.07). The proxy orders the K=6 neighbourhood correctly; at K=8 the reference schedule is retained.

Weight-variance bootstrap (Section 3.5). Table 29 bootstraps the fitted-weight standard deviation over 20 with-replacement resamples of the calibration set, at three calibration sizes. The spread falls as $P^{-1/2}$ (from 0.0680 at P=5 to 0.0399 at P=20, a 1.70× reduction), matching the proposition; the sharpness gain of calibration is real but, as the per-P sharpness of Table 25 shows, not monotone in P.

Operator spectrum (gap bound). Table 30 reports the spectral radius of the companion matrix induced by the fitted weights as a function of the forecast-window length L (the number of computed steps feeding a single forecast, i.e. the history size m). Here L is a *window* length, *not* the step gap $g(\mathcal{S})$ of Eq. 4, which bounds the spacing between consecutive computed steps; the two quantities appear in different tables. For $L \leq 7$ the radius stays within 3.5% of 1—numerically neutral, so a bounded skip run neither amplifies nor damps what it carries; at L=8 it jumps to 1.37, where the fit extrapolates rather than interpolates. Note that Eq. 2 is a one-shot projection from the computed steps, not a closed recursion—skipped outputs are never appended to the history—so this spectrum diagnoses the geometry of the fitted weights rather than an executed feedback loop. It is the empirical basis for capping the gap in Eq. 4.

Local Lipschitz constant ρ (skip-run envelope). The accumulation bound of Section 3.5 depends on the local Lipschitz constant ρ of the solver update of Eq. 1 composed with the denoiser, This quantity is measured directly on real PixArt-α DDIM trajectories (20 steps, CFG 4.5, 512×512). For each of 12 validation prompts we perturb the output of a single transformer block

Table 27: Closed-form schedule search on COCO val2017. For each K we score 300 randomly sampled gap-bounded schedules by the proxy residual $e(\mathcal{S})$ (Section 3.4); no images are rendered. The chosen schedules of Table 33 sit near the best of their K band. Over the six well-spread operating points e correlates with sharpness (Spearman -0.89, Kendall $\tau_b = -0.73$, n=6) but not with alignment (Spearman 0.49). The ranking holds within the gap-bounded, well-spread family that (G1)–(G3) generate and does not extend to schedules that cluster computed steps (Figure 6).

K	Scored	Best $e(\mathcal{S})\downarrow$	Mean $\pm$ std	Worst $e(\mathcal{S})$
6	300	0.2358	0.3298 ± 0.0389	0.4476
7	300	0.1838	0.2773 ± 0.0528	0.4331
8	300	0.1262	0.3408 ± 0.1546	0.9143

Table 28: Zero-extra-generation schedule selection. The proxy $e(\mathcal{S})$ (Section 3.4) places the guided schedules; each is then rendered once. References are the calibrated schedules of main-text Table 1. The K=6 guided point lands between the K=6 and K=7 reference points on both benchmarks (proxy prediction confirmed); the K=8 guided point is below the reference K=8 point, which is retained as the operating point.

Dataset	Schedule (K, steps)	Proxy e	s/img	Speedup ↑	CLIP ↑	Sharp. ↑
COCO	K6 guided $[0, 1, 3, 7, 12, 17]$	0.2453	2.70	3.05×	29.82	192.2
COCO	K6 reference $[0, 1, 2, 5, 9, 14]$	0.2847	3.328	2.48×	30.27	124.4
COCO	K7 reference $[0, 1, 2, 5, 9, 14, 18]$	0.1876	3.126	2.64×	30.16	225.8
MJHQ	K6 guided $[0, 1, 3, 7, 12, 17]$	0.2453	2.64	3.16×	29.95	190.6
MJHQ	K6 reference $[0, 1, 2, 5, 9, 14]$	0.2847	3.145	2.66×	30.12	126.3
COCO	K8 guided $[0, 1, 2, 4, 8, 13, 16, 18]$	0.1262	3.46	2.38×	29.77	382.7
COCO	K8 reference $[0, 1, 2, 4, 7, 11, 15, 18]$	0.1298	3.562	2.31×	30.07	480.2

$b \in \{0, 5, 11, 17, 23, 27\}$ at a denoising step $t \in \{8, 12, 16\}$ by δ of magnitude 0.01 of that block's output Frobenius norm, and measure the magnification of the *same* block's output at step $t{+}1$, $\rho_{t,b} = \|\mathrm{out}^{(b)}_{t+1}(\delta) - \mathrm{out}^{(b)}_{t+1}(0)\|/\|\delta\|$. Blocks are perturbed one at a time so the factor isolates the local map; because it is a norm ratio it is resolution-independent, so 512px suffices. Figure 7 summarizes the 216 measurements: the per-step magnification has median $\hat{\rho} = 0.45$ (per-block median 0.33–1.33), and 75% of samples have $\hat{\rho} < 1$—a skipped step typically *damps* the forecast error. The largest single-step magnification is 3.6 (block 27, early in the trajectory) and stays bounded. This is consistent with the neutral spectral radius of Table 30 for $L \leq 7$, and validates the gap-bounded schedule: under max gap ≤ 5 a skip run of length k multiplies the accumulated deviation by at most 3.6^k in the worst case and by $0.45^k \ll 1$ in the typical case, whereas an uncapped adaptive gate can produce longer runs whose bound grows geometrically without the cap.

E Alternative predictors and gates

This appendix documents the predictor and gate variants measured during the attribution study. Across all of them, added predictor expressivity does not improve alignment, and the variants that raise sharpness do so only by losing alignment, which is the evidence behind the attribution result of Section 4.5.

Higher-order fits. A degree-q polynomial extrapolation is the linear functional whose weights satisfy $\sum_k w_k u_k^j = t^j$ for $j \leq q$, so its leverage $h(t) = \phi(t)^\top (\Phi^\top \Phi)^{-1} \phi(t)$ measures how far the fit is asked to extrapolate. For q=1, $h(t) = \frac{1}{m} + \frac{(t-\bar{u})^2}{\sum_i (u_i - \bar{u})^2}$ and stays within $[0.6, 2.3]$ across the schedules we use; for q=2 it grows as $(t-\bar{u})^4$ and is 8–15× larger (e.g. 19.0 versus 2.3 at t=3 on the 8-step grid), and at $m \leq q$ the fit is underdetermined (a degree-q polynomial has $q{+}1$ free parameters). Measured on the calibration trajectory, the degree-2 extrapolation residual is 1.4–2.2× the affine residual at the same history length. The instability is thus intrinsic to the higher-order

Table 29: Bootstrap estimate of the fitted-weight standard deviation versus calibration size P on COCO val2017 (with-replacement, 20 replicates). The $P^{-1/2}$ decay matches Section 3.5.

P	Mean weight std ↓	Reduction vs P=5
5	0.0680	1.00×
10	0.0535	1.27×
20	0.0399	1.70×

Table 30: Spectral radius of the companion matrix induced by the fitted weights, versus forecast-window length L. For $L \leq 7$ the radius is within 3.5% of 1 (neutral); at L=8 it rises to 1.37 (amplifying), motivating the gap bound of Eq. 4.

L	Spectral radius	Regime
3	0.995	neutral
4	0.989	neutral
5	1.029	neutral
6	0.969	neutral
7	1.013	neutral
8	1.367	amplifying

fit on short, irregular windows rather than a property of any particular schedule—which is why the quadratic variant collapses in alignment (the main-text predictor-ablation table) even though its sharpness is high.

Least squares versus finite differences: a direct comparison. The preference for the least-squares form (Section 3) is a robustness choice, and we verify it directly against finite differences. On the same gap-bounded, well-spread schedules used in the paper, for every skipped step with $m \geq 2$ cached outputs we compare the affine least-squares forecast (ours) against a one-step finite-difference (FD1) and a two-step finite-difference (FD2) forecast, measuring the relative ℓ_2 residual of the predicted cached features against the true next cached output (zero generation; the collected calibration trajectories of Appendix C). Over 300 skipped steps the mean residual is 0.30 (LS), 0.24 (FD1) and 0.30 (FD2); the LS–FD1 gap is small and unsystematic—its correlation with the extrapolation ratio $r=(t-u_m)/(u_m-u_{m-1})$ is 0.05—and at m=2 the two are identical by construction (affine LS over two points *is* the line through them; $\max|\text{LS}-\text{FD1}|=0$). FD2 is worse at short histories (m=3–5: 0.11–0.17 versus 0.07–0.14) but the gap closes by m=6, consistent with the leverage argument of the previous paragraph. Two very different predictor forms thus produce nearly the same error on the same schedules, which is exactly the content of the attribution result of the main-text ablation study: the predictor's functional form is not the bottleneck, the schedule is. We retain the least-squares form because, as the main-text theoretical analysis shows, it is exact on affine observables for *any* spacing and its sensitivity is bounded by worst-case error independently of the local grid, whereas the finite-difference constants carry the schedule-dependent factor r whose tail we cannot bound without schedule-specific assumptions.

Per-block forecasting. Forecasting each transformer block from its own history introduces an independent error per block, and the 28 errors compose along the depth of the network; the resulting output is dominated by the accumulated error. Applying the same forecast once, to the output of the whole stack, avoids this composition, which is why our method forecasts at the stack level.

The depth–time hybrid. Recomputing the first k blocks at every step and forecasting the remaining residual does inject a fresh signal, and the sharpness of the resulting images does rise (from 63.8 to 93.1 for $k = 1$ and to 133.3 for $k = 2$ on COCO). Both configurations are nevertheless dominated: their CLIP score is no better than the plain forecast at the same schedule while their wall-clock time is higher, and the token-space variant that forecasts the residual rather than holding it verbatim produced high-frequency artefacts.

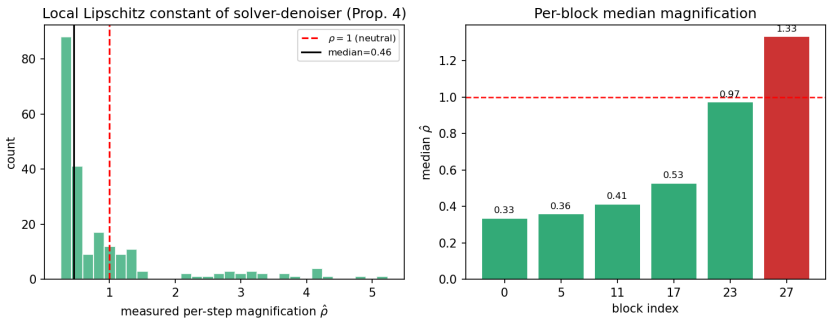


Figure 7: Measured per-step magnification $\rho_{t,b}$ of a perturbed transformer-block output across 12 prompts and six blocks. Histogram (left) and per-block medians (right); the whiskers show the interquartile range. The majority of perturbations are contractive ($\rho < 1$), confirming that a skipped step usually damps rather than amplifies the forecast error.

Table 31: Predictor and gate variants at the fixed 8-step schedule, COCO val2017 (gating rows at their own operating points). "Artefacts" means the Laplacian statistic rose while CLIP dropped, with high-frequency artefacts on visual inspection rather than recovered detail.

Direction	Mechanism	CLIP ↑	Outcome
Per-block forecasting	forecast each of 28 blocks separately	19.42	not adopted (error compounds)
Quadratic least squares	degree-2 polynomial fit	26.18	not adopted
Predictor–corrector feedback	decayed correction of the systematic error	29.21	not adopted
Recency-weighted least squares	exponential weighting of history	30.04	no gain over affine
Clean-signal reparameterization	forecast $(x_t - \sigma_t f_t)/\alpha_t$	20.46	not defined for this checkpoint
Adaptive gating (τ=0.04)	first-order statistic on the input	22.27	not adopted
Adaptive gating (τ=0.05)	first-order statistic on the input	20.34	not adopted
Adaptive gating (τ=0.07)	first-order statistic on the input	19.53	not adopted
Depth–time hybrid, forecast residual	recompute k blocks, forecast the rest	artefacts	not adopted
Depth–time hybrid, verbatim residual	recompute k blocks, cache the rest	29.86	dominated by the 6-step schedule

The clean-signal reparameterization. The reparameterization writes the network output as $\epsilon = (x_t - \alpha_t x_0)/\sigma_t$ and forecasts the more stationary quantity x_0. This requires the network output to live in the same space as the latent $\boldsymbol{x}_t$. In the checkpoint we use, the transformer emits an 8-channel tensor while the latent has 4 channels, so the algebraic inversion is not defined for this checkpoint.

F Learning-to-Cache under our protocol

Learning-to-Cache (L2C, (Ma et al., 2024a)) learns, per denoising step, which of a transformer's submodules may reuse the output of the previous step. It requires training a router, so we report it here rather than in the training-free comparison of main-text Table 1.

Protocol. L2C ships routers for DiT-XL/2 and U-ViT-H/2 on 256px ImageNet; we train a router for PixArt with the authors' objective and code (`Router` and `STE`), on COCO val2017 caption pairs for 6,200 steps. Their $2\times$ *depth*-output router is mapped onto PixArt's self-attention and feed-forward with cross-attention always computed, and their continuous-timestep bookkeeping is adapted to our 20 discrete DDIM steps. As with ToCa, with every router weight forced to 1 (nothing reused) the grafted pipeline is bit-identical to the uncached one.

Results. Table 32 is the speed–quality curve on COCO. Dropping 9% of the submodule computations leaves CLIP at 30.05 (indistinguishable from 30.00 uncached) for $1.04\times$, while dropping 49% takes CLIP to 18.85 for $1.55\times$; masking a *random* 49% of the same submodules gives 18.57,

within noise of the learned policy. The usable operating point on this backbone is $1.04\times$, matching TGATE.

Table 32: L2C on PixArt-α, 512px, DDIM-20, CFG 4.5, COCO val2017. "Dropped" is the fraction of self-attention/feed-forward computations replaced by the previous step's output. The last row replaces the learned router by a random mask with the same keep-rate.

Setting	Dropped	s/img	Speedup	CLIP ↑
Uncached	0.00	8.05	1.00×	30.00
L2C, θ=0.20	0.09	7.75	1.04×	30.05
L2C, θ=0.50	0.49	5.21	1.55×	18.85
L2C, θ=0.60	0.65	4.00	2.01×	19.50
Random mask, keep 0.51	0.49	5.05	1.60×	18.57

G COMPUTED-STEP SCHEDULES

Table 33: Computed-step schedules used in this paper. On the 20-step PixArt solver the first three steps are always computed, since no forecast is possible before two outputs exist; on the 28-step SD3.5 trajectory we anchor the first two steps and space the rest, which is why its schedules start $\{0, 1, 3, \dots\}$ or $\{0, 1, 5, \dots\}$. All schedules satisfy the gap bound of Eq. 4

Label	$K = \lvert\mathcal{S}\rvert$	Computed steps $\mathcal{S}$
6-step	6	$\{0, 1, 2, 5, 9, 14\}$
7-step	7	$\{0, 1, 2, 5, 9, 14, 18\}$
8-step	8	$\{0, 1, 2, 4, 7, 11, 15, 18\}$
9-step	9	$\{0, 1, 2, 4, 6, 9, 12, 15, 18\}$
SD3.5-Medium, 28-step flow-matching trajectory		
6-step	6	$\{0, 1, 7, 14, 20, 27\}$
7-step	7	$\{0, 1, 5, 10, 16, 22, 27\}$
8-step	8	$\{0, 1, 3, 6, 10, 15, 20, 25\}$

Reproducing the schedules. We verify the guideline of Section 3.4 against the schedules actually used. Every schedule in Table 33 (and the two residual-guided schedules of Appendix D.5) contains $\{0, 1, 2\}$ (G1) and obeys the gap bound (G2): the maximal gap is 5 for the 6- and 7-step schedules, 4 for the 8-step and 3 for the 9-step schedule, all $\leq g_{\max} = 5$. For (G4), each proxy sits at or near the front of the 300-sample search of Table 27 in its K band: the 7-step proxy (0.1876) is within 0.004 of the band best (0.1838) and the 8-step proxy (0.1298) within 0.004 of its band best (0.1262), while the chosen 6-step proxy (0.2847) prioritises an even spread over a minimal residual. The guideline therefore generates the feasible set and the proxy selects near the front of it.

H PROOFS

Proof of Proposition 1 (exactness on polynomial observables). *Let $u_1 < \cdots < u_m$ be computed step indices and let $\hat{f}_t$ be given by Eq. 2 with $q = 1$. If the observable is affine in the step index, i.e. $f(u) = a + bu$, then $\hat{f}_t = f(t)$ for every t, independently of the spacing of the u_i.*

Proof. If $f(u_i) = a + bu_i$ for all i, the residual of the linear model $\Phi\mathbf{a}$ is zero, so the least-squares solution satisfies $\mathbf{f} = \Phi\mathbf{a}$ exactly. The prediction is $\hat{f}_t = [1, u_t]\,\mathbf{a} = \alpha + \beta\, u_t = f(t)$ (we write $\mathbf{a} = (\alpha, \beta)$ for the polynomial parameters to avoid clashing with the history weights w). The argument does not involve the spacing, whereas a finite-difference scheme of the same order estimates b from the difference of the two nearest points and divides by $u_m - u_{m-1}$, so its prediction error is $\mathcal{O}((t - u_m)/(u_m - u_{m-1}))$ times the residual and can therefore grow without bound as the last gap shrinks while the extrapolation distance grows. □

Proof of Proposition 2 (sensitivity of the forecast). *The weights of Eq. 2 satisfy* $\sum_i w_i(t) = 1$ *and* $\|w(t)\|_1 \le 1 + |u_t - \bar{u}| \sum_i |u_i - \bar{u}|/S_2$.

Proof. Summing Eq. 2 over i gives $\sum_i w_i(t) = 1 + \frac{t-\bar{u}}{S_2} \sum_i (u_i - \bar{u}) = 1$, because $\sum_i (u_i - \bar{u}) = 0$. The ℓ_1 bound is the triangle inequality applied to the same expression. Since the forecast is the linear functional $\hat{f}_t = \sum_i w_i(t) f_{s_i}$, a perturbation of the cached outputs of size δ produces $\|\delta \hat{f}_t\| \le \|w(t)\|_1 \, \delta$. For m equally spaced computed steps with span $L = u_m - u_1$ one has $\sum_i |u_i - \bar{u}|/S_2 \to 3/L$, so the amplification is governed by the extrapolation distance measured in units of the spread of the computed steps; this is the quantitative form of the statement that spreading the computed steps—rather than clustering them—protects the forecast. □

Measured check. Bootstrapping the calibration set (resampling trajectories with replacement, 20 replicates) gives a mean standard deviation of the fitted weights of 0.068 at P=5, 0.054 at P=10 and 0.040 at P=20, close to the $P^{-1/2}$ rate the variance-reduction statement of Section 3.5 predicts (Table 29). The reduction in estimator variance is therefore real and measurable. The sharpness statistic is not monotone in P (Table 25); we report the variance reduction as a property of the estimator, not as a complete account of the sharpness gain.

Proof of Proposition 3 (variance reduction by calibration). *Let* $\hat{w}^{(m)}_{img}$ *be the least-squares weights estimated from a single trajectory and* $\hat{w}^{(m)}_{cal}$ *those from Eq. 3 over* P *trajectories, with independent, mean-zero observation noise of variance* σ^2. *Then both are unbiased for the same population weights* $w^{(m)}$, *and* $\mathrm{Var}(\hat{w}^{(m)}_{cal}) = P^{-1} \mathrm{Var}(\hat{w}^{(m)}_{img})$.

Proof. Both estimators solve a normal-equation system with the same design matrix Φ_m formed from the step indices, which are identical across trajectories because the schedule is fixed. The single-trajectory estimator has covariance $\sigma^2(\Phi_m^\top \Phi_m)^\dagger$; pooling P independent trajectories stacks the design matrix P times, giving $P\,\Phi_m^\top \Phi_m$ and hence the stated covariance. Unbiasedness follows from $\mathbb{E}[\mathbf{f}] = \Phi_m w^{(m)}$. □

Remark. The proposition covers estimator variance, not approximation error: if the trajectory is not affine, both estimators share the same bias. This is precisely why the quadratic fit reported in Appendix E does not help—it reduces bias in principle but inflates estimator variance, which dominates at our history lengths.

Proof of Proposition 4 (error accumulation over a skip run). *Suppose each prediction introduces an error bounded by* ϵ *in the network output, and suppose that a perturbation* δ *in the output at step* t *results in a perturbation of at most* $\rho\|\delta\|$ *in the output at step* $t-1$ *(local Lipschitz constant* ρ *of the solver–denoiser composition). Then after* k *consecutive skipped steps the accumulated deviation is at most* $\epsilon(\rho^k - 1)/(\rho - 1)$ *for* $\rho > 1$, *and at most* ϵk *for* $\rho \le 1$.

Proof. By induction. Let Δ_j bound the deviation after j skips. The deviation entering skip $j+1$ is amplified by at most ρ through Eq. 1 and the denoiser, and a fresh prediction error of at most ϵ is added: $\Delta_{j+1} \le \rho\,\Delta_j + \epsilon$, with $\Delta_0 = 0$. Unrolling gives the stated bound. □

Remark. The bound is geometric in the length of the longest skip run, which is the formal reason we constrain the maximal gap in Eq. 4 rather than only the total number of computed steps. It also explains why an adaptive gate that occasionally produces very long runs—as we observed at $\tau = 0.07$, where the effective skip rate rose to 0.80—degrades quality sharply, while schedules with an explicit gap bound do not.