KALMANCACHE+: UNCERTAINTY-AWARE CLOSED-LOOP DIFFUSION CACHING VIA PREDICT–TRUST–CORRECT RESIDUAL ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion transformers (DiTs) achieve high-fidelity visual generation but incur substantial inference costs from repeated block evaluations across denoising steps. Training-free caching mitigates these costs through feature or residual reuse, yet fixed schedules, local proxies, and model-specific calibration provide limited control over cache reliability and accumulated skip errors. We introduce *KalmanCache+*, an uncertainty-aware closed-loop framework that formulates diffusion caching as sequential residual estimation. A lightweight channel-wise Kalman filter predicts residual evolution and uses predictive uncertainty to choose between FULL computation and SKIP. Prediction errors observed at FULL steps refine subsequent decisions and residual reconstructions during SKIP steps. This Predict–Trust–Correct cycle adapts online to each sampling run without modifying pretrained weights or requiring calibration tables. Across image and video DiTs, *KalmanCache+* achieves latency speedups of $1.91\times$ on DiT-XL/2 (FID 2.241 vs. 2.282 for full computation), $3.10\times$ on FLUX.1-dev (LPIPS 0.199 vs. 0.368 for step reduction at $2.88\times$), and $2.13\times$ on Wan2.1-1.3B (PSNR 29.55 vs. 23.16 dB for TeaCache at $2.05\times$).

1 INTRODUCTION

Diffusion transformers (DiTs) have emerged as the dominant paradigm for high-fidelity image and video synthesis, establishing state-of-the-art performance in visual quality, text-image alignment, and long-range temporal coherence (Peebles & Xie, 2023; Esser et al., 2024; Wan et al., 2025). This quality comes at a high inference cost: a single sample takes tens of sequential denoising steps, and each step evaluates the full transformer over a token sequence that grows with resolution and frame count (Peebles & Xie, 2023; Wan et al., 2025). The resulting latency and GPU cost are a major bottleneck for interactive generation and high-throughput server-scale deployment.

Existing acceleration methods either reduce the number of network evaluations or lower the cost of each. Step-reduction samplers such as DDIM and DPM-Solver take fewer, larger steps but alter the sampling trajectory, so aggressive settings degrade sample quality (Song et al., 2021; Lu et al., 2022; Zhao et al., 2023). Distillation trains a few-step student from a many-step teacher, but requires costly additional training for each model (Salimans & Ho, 2022; Song et al., 2023). Pruning removes redundant channels but requires sparsity training or fine-tuning (Kim & Kim, 2024; Fang et al., 2023); quantization lowers numerical precision at inference or during training but needs calibration data or custom low-precision kernels (Kim et al., 2024; Choi & Kim, 2025; Wu et al., 2024); and token merging shortens the token sequence without training but can lose fine detail at high merging ratios (Bolya & Hoffman, 2023). Training-free caching exploits the redundancy between nearby denoising steps: it reuses or forecasts features from earlier steps, leaving the sampler and pretrained weights unchanged (Ma et al., 2024b; Selvaraju et al., 2024; Liu et al., 2025a;b). This plug-and-play property has made caching an important direction for accelerating pretrained diffusion models.

Caching methods make two key choices: which steps to compute in full (FULL steps) and what to use at the skipped ones (SKIP steps). Some recompute at a fixed interval or on a hand-designed or learned schedule (Selvaraju et al., 2024; Zou et al., 2025; 2026; Liu et al., 2025b; Chen et al., 2024; Ma et al., 2024a). Others recompute once a change proxy accumulated since the last full computation

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.

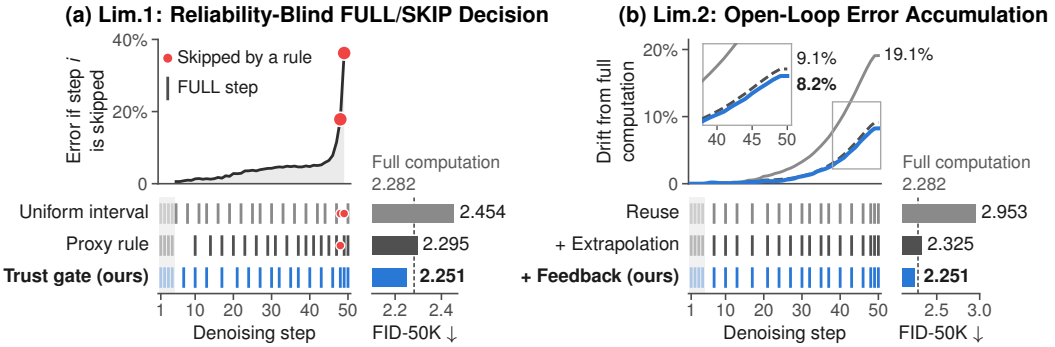


Figure 1: Where caching loses quality and how *KalmanCache+* recovers it on DiT-XL/2 (ours: one trust-gate schedule replayed on all samples; adaptive gate: FID 2.241).

exceeds a threshold, where the proxy is measured online (Liu et al., 2025a; Bu et al., 2026) or read from a curve calibrated offline (Ma et al., 2025). At the skipped steps, most of them reuse the cached features or residuals, while some forecast them from recent full computations (Liu et al., 2025b; Bu et al., 2026). However, these caching methods have limitations in both respects.

(Lim.1) Reliability-Blind FULL/SKIP Decision. Neither a schedule nor a change proxy estimates how uncertain the cached prediction is, so either rule can skip exactly the steps where the cache is least reliable. Figure 1(a) illustrates this on DiT-XL/2 at a budget of about 21 of 50 FULL steps, with the SKIP reconstruction held fixed: the error of reusing the preceding step's residual, measured on full-computation trajectories, stays below 12% through step 47 and jumps to 18% and 36% at steps 48 and 49. The uniform interval (every 2–3 steps) skips both, and an input-change-derived schedule (proxy rule) built on the signal of TeaCache (Liu et al., 2025a) skips step 48; their FID-50K is 2.454 and 2.295 against 2.282 for full computation.

(Lim.2) Open-Loop Error Accumulation. Whether they reuse or forecast, the compared caching methods use a FULL step only to refresh the cache: the mismatch between their prediction and the computed residual is not fed back to correct later SKIP steps, so every SKIP-step error is carried forward by the sampler. Figure 1(b) traces this with the FULL/SKIP schedule held fixed: with plain reuse, the latent drifts steadily away from the full-computation trajectory and ends 19.1% away (relative ℓ_2), and extrapolating the residual approximately halves this drift, giving FID-50K 2.953 and 2.325 against 2.282 for full computation.

To address these limitations, we formulate training-free diffusion caching as an uncertainty-aware sequential residual estimation problem and propose *KalmanCache+*. For (Lim.1), it tracks a lightweight channel-wise state of the residual and its uncertainty with a Kalman filter (Kalman, 1960) and, outside the warmup and the final step, skips only while the predicted uncertainty stays below a trust threshold, so every such decision rests on an explicit uncertainty estimate. For (Lim.2), each gate-triggered FULL step provides two forms of feedback: the observed channel means update the state and its noise statistics, and the observed tensor-level mismatch corrects the reconstruction of the next SKIP segment. *KalmanCache+* reaches a 1.91× latency speedup on DiT-XL/2 with FID 2.241 against 2.282 for full computation, 3.10× on FLUX.1-dev with LPIPS 0.199 against 0.368 for step reduction at 2.88×, and 2.13× on Wan2.1-1.3B with PSNR 29.55 dB against 23.16 dB for TeaCache at 2.05×. Our main contributions are summarized as follows:

- *Uncertainty-Aware Closed-Loop Diffusion Caching:* We cast caching as sequential residual estimation in which each FULL step is a costly observation, and propose the Predict–Trust–Correct cycle: predicted uncertainty decides when to observe, and each observation corrects later decisions and SKIP reconstructions.
- *Predict–Trust (solution to Lim.1):* A lightweight state-space estimator tracks channel-wise states and covariances of the residual, and a covariance-based trust gate selects FULL or SKIP, adapting to each run without a fixed schedule.
- *Correct (solution to Lim.2):* The mismatches observed at FULL steps are fed back into later trust decisions through online noise adaptation and into SKIP reconstructions through a compensation–disturbance observer (C-DOB), without model-specific offline calibration tables.

2 Background and Related Work

2.1 Diffusion Caching as Sequential Residual Estimation

Diffusion and flow-based generative models turn random noise into a sample over T sampling steps (Ho et al., 2020; Song et al., 2021; Liu et al., 2023), indexed by $i = 1, \ldots, T$. At each step, a denoising network maps the noisy latent and the condition (*e.g.*, a class label or a text prompt) to its target (*e.g.*, noise, score, or velocity). In DiTs this network is a deep stack of transformer blocks whose cost grows with spatial resolution and frame count (Peebles & Xie, 2023; Wan et al., 2025).

Caching exploits the redundancy between neighboring steps by replacing some module evaluations with cached values or predicted updates. Let m denote a cacheable module, $z_i^{(m)}$ its input at step i, and $y_i^{(m)}$ its full-computation output. We define the residual tensor, the SKIP output, and the residual approximation error as

$$\mathcal{R}_i^{(m)} = y_i^{(m)} - z_i^{(m)}, \quad \widehat{y}_i^{(m)} = z_i^{(m)} + \widehat{\mathcal{R}}_i^{(m)}, \quad e_i^{(m)} = \mathcal{R}_i^{(m)} - \widehat{\mathcal{R}}_i^{(m)}. \tag{1}$$

At a FULL step, module m is evaluated and its exact residual $\mathcal{R}_i^{(m)}$ is observed; at a SKIP step $\widehat{y}_i^{(m)}$ is rebuilt from an approximate residual $\widehat{\mathcal{R}}_i^{(m)}$ estimated from previous steps. Cache quality is thus governed by $e_i^{(m)}$, and caching is not mere feature reuse but a sequential estimation problem: it must assess the reliability of the current approximation and feed FULL-step residuals into later predictions.

Kalman filtering (Kalman, 1960) is a classical framework for such problems: a predict step propagates a state estimate and its error covariance, and an update step corrects the estimate with the innovation, the difference between an observation and its prediction. *KalmanCache+* follows this structure: its Predict and Correct stages build on the filter's predict and update steps, respectively. Since each residual observation costs a FULL evaluation, a Trust stage between them decides when to observe, choosing FULL or SKIP at each step outside the initial warmup steps and the final step.

2.2 Compared Caching Methods: Reliability-Blind and Open-Loop

The compared schedule-, proxy-, and forecasting-based methods decide FULL/SKIP without quantifying the uncertainty of their residual prediction. Layer- or token-level methods reuse DiT activations on fixed or learned schedules (Selvaraju et al., 2024; Chen et al., 2024; Zou et al., 2025; 2026; Ma et al., 2024a), and a schedule fixed before sampling cannot judge whether a cached value is reliable at a given step. Proxy-signal methods accumulate a scalar change proxy since the last FULL step, such as an input difference (Liu et al., 2025a), a shallow-probe difference (Bu et al., 2026), or a magnitude ratio from a calibrated curve (Ma et al., 2025), and refresh once it exceeds a threshold, but this proxy is a point estimate of change or error, not the uncertainty of the cached value. Trajectory-forecasting methods extrapolate features on a fixed schedule without a runtime estimate of the forecast error (Liu et al., 2025b). This is the reliability-blind FULL/SKIP decision problem of **(Lim.1)**.

On the correction side, none of the compared methods feeds back the mismatch between its own prediction and the residual observed at a later FULL step, so the error of each SKIP step is carried forward by the sampler. Calibration-dependent methods additionally precompute a rescaling polynomial or a magnitude curve from reference runs, a reasonable prior that assumes new inputs share the residual dynamics of those runs (Liu et al., 2025a; Ma et al., 2025), but do not adjust it at runtime when this assumption fails. This is the open-loop error accumulation problem of **(Lim.2)**.

3 Proposed Method: KalmanCache+

3.1 Overview

KalmanCache+ couples a lightweight channel-wise Kalman filter with the residual cache in a Predict–Trust–Correct cycle (Figure 2): Predict estimates how the residual has evolved since the last FULL step and how uncertain this is, Trust uses this uncertainty to decide whether to skip, and Correct turns each FULL computation into feedback for later predictions and SKIP reconstructions. The channel-wise filter estimates residual dynamics for uncertainty-based control, while the full residual tensor is reconstructed from recent FULL residuals and a stored extrapolation mismatch.

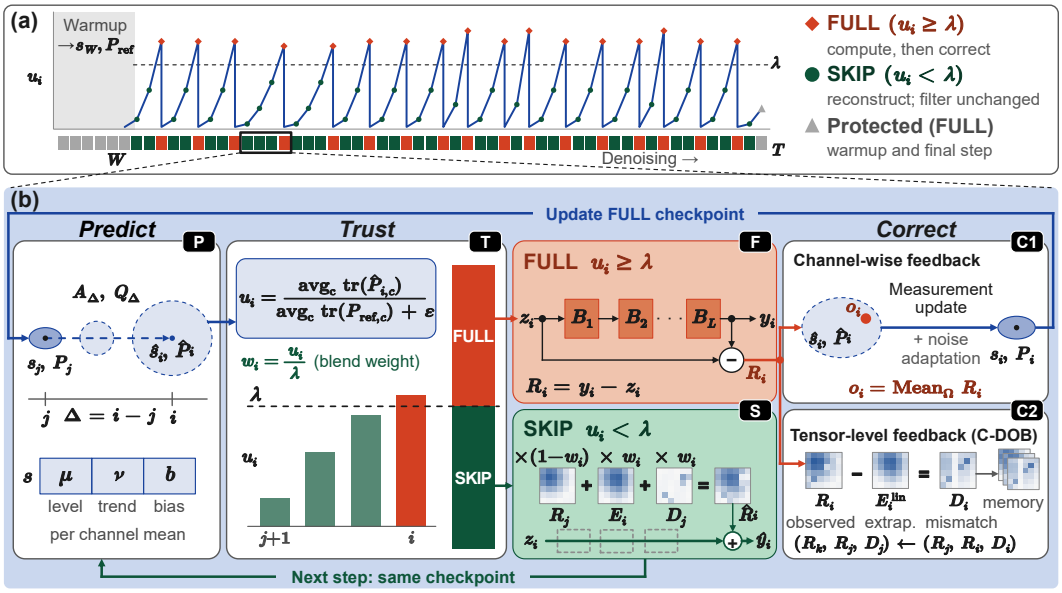


Figure 2: Uncertainty-aware closed-loop caching in *KalmanCache+*. (a) FULL/SKIP decisions over a sampling run. (b) The Predict–Trust–Correct cycle between two FULL steps.

Over a sampling run (Figure 2(a)). The first W steps (warmup) and the final step T are protected and always run FULL; the warmup residuals initialize the state s_W and the reference covariance P_{ref}. Afterwards, the uncertainty score u_i of the trust gate typically rises along each SKIP segment and each FULL correction can lower it (schematic sawtooth); as the filter starts from this run's warmup and re-estimates its noise scales at each later FULL step, FULL-step spacing adapts to each run.

The cycle between two FULL steps (Figure 2(b)). Let $k < j$ be the last two FULL steps before an unprotected step i (j+1 to i boxed in Figure 2(a)). **P** Predict takes the FULL checkpoint (s_j, P_j) (solid ellipse), the estimated level, trend, and bias of each residual channel mean with their uncertainty, and propagates it over the gap $\Delta = i - j$, with transition A_Δ and process noise Q_Δ, to the prediction $(\hat{s}_i, \hat{P}_i)$; the ellipses sketch the accumulating process noise. **T** Trust condenses this uncertainty $\hat{P}_i$ into the score u_i, which typically rises from j+1 to i (bars): below the trust threshold λ (dashed line) the step is skipped, and once u_i reaches λ (red bar) it runs FULL. Trust also passes the blend weight $w_i = u_i/\lambda$ to SKIP. **S** SKIP bypasses the blocks (dashed) and outputs $\hat{y}_i = z_i + \hat{\mathcal{R}}_i$, blending the cached residual $\mathcal{R}_j$, its extrapolation $\mathcal{E}_i$, and the stored mismatch D_j with weights $1 - w_i$, w_i, w_i; the filter is unchanged (*Next step: same checkpoint*). **F** FULL runs the blocks $B_1, \ldots, B_L$ and passes the observed residual $\mathcal{R}_i = y_i - z_i$ to Correct. **C1** *Channel-wise feedback* uses the channel means o_i of $\mathcal{R}_i$ to update $(\hat{s}_i, \hat{P}_i)$, with noise adaptation, into a less uncertain (s_i, P_i), the new checkpoint (*Update FULL checkpoint*), and **C2** *tensor-level feedback (C-DOB)* measures the mismatch D_i between $\mathcal{R}_i$ and its extrapolation $\mathcal{E}_i^{\text{lin}}$ and overwrites the residual memory $(\mathcal{R}_k, \mathcal{R}_j, D_j)$ with $(\mathcal{R}_j, \mathcal{R}_i, D_i)$ for the next SKIP segment. This closed loop is the contribution; the estimators and variance-based triggering themselves are classical (Kalman, 1960; Sage & Husa, 1969; Trimpe & D'Andrea, 2014).

3.2 Predict–Trust: Uncertainty-Aware FULL/SKIP Decision

Predict–Trust is our solution to **(Lim.1)**: every gated FULL/SKIP decision reads an explicit, gap-dependent estimate of the residual prediction's uncertainty, not a fixed schedule or a change proxy.

Predict. For each of the C channels c of the cacheable module m (the transformer-block stack), the filter tracks the channel mean of the residual with a state estimate and its error covariance:

$$s_{i,c}^{(m)} = \left[\mu_{i,c}^{(m)},\ \nu_{i,c}^{(m)},\ b_{i,c}^{(m)}\right]^\top, \qquad P_{i,c}^{(m)} \in \mathbb{R}^{3\times 3}. \tag{2}$$

Between FULL steps, the filter predicts where each channel mean is heading: the level μ is the channel mean, the trend ν its change per sampling step, and the bias b a persistent offset that trend

extrapolation alone would miss, so the level expected Δ steps later is $\mu + \Delta\nu + b$; P records the uncertainty of all three. Tracking channel means instead of the full tensor keeps the filter small. Predict propagates the checkpoint of the last FULL step j over the gap $\Delta_i = i - j$:

$$\widehat{s}^{(m)}_{i|i-1,c} = A_{\Delta_i}\, s^{(m)}_{j|j,c}, \qquad \widehat{P}^{(m)}_{i|i-1,c} = A_{\Delta_i}\, P^{(m)}_{j|j,c}\, A^{\top}_{\Delta_i} + Q^{(m)}_{\Delta_i,c}, \tag{3}$$

where $j|j$ denotes the estimate after the update at step j and $i|i-1$ the prediction for step i given all earlier FULL observations. Predict passes $\widehat{P}$ to Trust and $(\widehat{s}, \widehat{P})$ to Correct.

Trust. Trust condenses the predicted covariances into the uncertainty score $u^{(m)}_i$, the channel-averaged trace of $\widehat{P}^{(m)}_{i|i-1,c}$ relative to that of the reference covariance $P^{(m)}_{\mathrm{ref},c} = P^{(m)}_{W,c}$ at the end of warmup, and its trust gate allows SKIP only while $u^{(m)}_i$ stays below the trust threshold λ:

$$u^{(m)}_i = \frac{\frac{1}{C}\sum_{c=1}^{C} \mathrm{tr}\Big(\widehat{P}^{(m)}_{i|i-1,c}\Big)}{\frac{1}{C}\sum_{c=1}^{C} \mathrm{tr}\Big(P^{(m)}_{\mathrm{ref},c}\Big) + \varepsilon}, \qquad d^{(m)}_i = \begin{cases} \text{FULL}, & i \in \mathcal{P} \text{ or } u^{(m)}_i \geq \lambda, \\ \text{SKIP}, & \text{otherwise.} \end{cases} \tag{4}$$

Here, $\mathcal{P} = \{1, \ldots, W\} \cup \{T\}$ is the protected-step set and ε a small constant for numerical stability. Scoring each step this way matters for caching because the reliability of a residual prediction changes within a run (Appendix C). For unprotected steps on the image backbones, λ is a relative tolerance: the cache is refreshed once the predicted uncertainty reaches λ times its level at the end of warmup, and through w_i the same λ sets how each SKIP step is reconstructed. For video, the gate compares $u^{(m)}_i$ with $\lambda\rho$, where $\rho \in [0, 1]$ is an online autocorrelation estimate of consecutive FULL channel-mean residuals, so weaker autocorrelation triggers FULL sooner. Like the interval, threshold, or step count of the compared methods, λ is set offline per configuration, with a backbone-specific scale; the distinction lies in what the decision reads, not in whether a control parameter is set offline. Trust outputs $d^{(m)}_i$ and, for SKIP, the blend weight $w^{(m)}_i = u^{(m)}_i/\lambda$.

Trust's decision is then executed: a SKIP step bypasses the blocks and reconstructs the residual from the residual memory, which holds the last two FULL residuals $\mathcal{R}^{(m)}_k$ and $\mathcal{R}^{(m)}_j$ ($k < j$) and the mismatch $D^{(m)}_j$ stored by the C-DOB, as a blend weighted by $w^{(m)}_i$:

$$\begin{aligned} \mathcal{E}^{\mathrm{lin}\,(m)}_i &= \mathcal{R}^{(m)}_j + \tfrac{i-j}{j-k}\big(\mathcal{R}^{(m)}_j - \mathcal{R}^{(m)}_k\big), \\ \widehat{\mathcal{R}}^{(m)}_i &= \big(1 - w^{(m)}_i\big)\, \mathcal{R}^{(m)}_j + w^{(m)}_i\, \mathcal{E}^{(m)}_i + w^{(m)}_i\, D^{(m)}_j, \qquad w^{(m)}_i = u^{(m)}_i/\lambda, \end{aligned} \tag{5}$$

where $\mathcal{E}^{(m)}_i = \mathcal{E}^{\mathrm{lin}\,(m)}_i$ for the image backbones; for video, $\mathcal{E}^{(m)}_i$ is $\mathcal{E}^{\mathrm{lin}\,(m)}_i$ with its second term scaled by ρ^{Δ_i}, while $w^{(m)}_i$ keeps λ, not $\lambda\rho$. While $u^{(m)}_i$ is small, the step mostly reuses $\mathcal{R}^{(m)}_j$; as $w^{(m)}_i$ grows, it shifts toward the corrected extrapolation $\mathcal{E}^{(m)}_i + D^{(m)}_j$, so the same uncertainty decides whether and how a step is skipped. SKIP outputs $\widehat{y}^{(m)}_i = z^{(m)}_i + \widehat{\mathcal{R}}^{(m)}_i$ and leaves the filter unchanged; FULL runs the blocks and passes $\mathcal{R}^{(m)}_i = y^{(m)}_i - z^{(m)}_i$ to Correct.

3.3 Correct: Closed-Loop Residual Feedback

Correct is our solution to **(Lim.2)**: instead of only refreshing the cache, it feeds the residual $\mathcal{R}^{(m)}_i$ that each post-warmup FULL step observes at the current, possibly cache-perturbed, input back into the filter and the residual memory.

Channel-wise feedback: a new checkpoint for Predict. The filter observes the channel means of the residual,

$$o^{(m)}_{i,c} = \mathrm{Mean}_{\Omega}\Big(\mathcal{R}^{(m)}_{i,c}\Big), \tag{6}$$

where Ω is the set of non-channel indices (spatial, token, and temporal, plus the sample and guidance indices for batched sampling). This is the filter's only measurement: it observes the level μ, and the trend and bias are corrected through their covariance with it. The innovation, $o^{(m)}_{i,c}$ minus the predicted level, is Predict's error: the Kalman update corrects the state with it and shrinks the predicted covariance, and online Sage–Husa estimation (Sage & Husa, 1969) re-estimates the measurement-

and process-noise scales from it, the former absorbing the mismatch between the channel-mean observations and the simplified state dynamics. This adaptation is how FULL observations reach later uncertainty scores: with fixed noise scales and FULL-step locations, the scores would not depend on the residuals observed after warmup. The corrected state and covariance, with the adapted noise scales, become the new FULL checkpoint for Predict (*Update FULL checkpoint*).

Tensor-level feedback (C-DOB): a refreshed memory for SKIP. From the third FULL step on (warmup included), the *compensation–disturbance observer* (C-DOB) treats the linear extrapolation from the two preceding FULL residuals as its nominal model and, like a disturbance observer, estimates what this model misses in each observed residual:

$$D_i^{(m)} = \mathcal{R}_i^{(m)} - \mathcal{E}_i^{\mathrm{lin}\,(m)}. \tag{7}$$

The mismatch $D_i^{(m)}$ is not the true error of an unobserved SKIP step, but a tensor-level extrapolation error directly measurable at a FULL step. The memory then shifts from $(\mathcal{R}_k^{(m)}, \mathcal{R}_j^{(m)}, D_j^{(m)})$ to $(\mathcal{R}_j^{(m)}, \mathcal{R}_i^{(m)}, D_i^{(m)})$, and each step of the next SKIP segment adds this mismatch with its own weight (Eq. 5).

The cycle in summary. What each stage reads and passes on (Algorithm 1, Appendix B):

$$(s_j, P_j) \xrightarrow{\text{Predict}} (\widehat{s}_i, \widehat{P}_i), \qquad \widehat{P}_i \xrightarrow{\text{Trust}} (d_i, w_i),$$

SKIP: $(z_i, w_i;\, \mathcal{R}_k, \mathcal{R}_j, D_j) \longrightarrow \widehat{y}_i$, checkpoint kept: step $i{+}1$ is predicted from (s_j, P_j),

FULL: $(\widehat{s}_i, \widehat{P}_i;\, \mathcal{R}_k, \mathcal{R}_j, \mathcal{R}_i) \xrightarrow{\text{Correct}} (s_i, P_i)$ to Predict, $(\mathcal{R}_j, \mathcal{R}_i, D_i)$ to SKIP.

A SKIP step thus hands the same checkpoint to the next prediction over a longer gap, and a FULL step hands Predict a new checkpoint and SKIP a refreshed memory, so every later decision and reconstruction uses the latest observation.

4 Experiments

4.1 Experimental Setup

Base Models and Compared Methods. We use three backbones, all with 50 sampling steps: class-conditional DiT-XL/2 (Peebles & Xie, 2023) with DDIM (Song et al., 2021) at 256×256, text-to-image FLUX.1-dev (Black Forest Labs, 2024) with rectified-flow (Liu et al., 2023) sampling at 1024×1024, and text-to-video Wan2.1-1.3B (Wan et al., 2025) with UniPC (Zhao et al., 2023) for 81 frames at 832×480. Besides step reduction, we compare with the caching methods L2C (Ma et al., 2024a), FORA (Selvaraju et al., 2024), ToCa (Zou et al., 2025), DuCa (Zou et al., 2026), Δ-DiT (Chen et al., 2024), TaylorSeer (Liu et al., 2025b), TeaCache (Liu et al., 2025a), DiCache (Bu et al., 2026), and MagCache (Ma et al., 2025). In the tables, N is a caching interval, O the TaylorSeer order, NFE the number of function evaluations, and l, δ, and K method-specific control parameters.

Evaluation Metrics. For DiT-XL/2, we follow the standard protocol of 50K samples with labels drawn uniformly from the 1,000 ImageNet (Deng et al., 2009) classes and report Fréchet inception distance (FID-50K) (Heusel et al., 2017), spatial FID (sFID) (Nash et al., 2021), Inception Score (IS) (Salimans et al., 2016), Precision, and Recall (Kynkäänniemi et al., 2019). For FLUX.1-dev, we use the 200 DrawBench (Saharia et al., 2022) prompts and report ImageReward (IR) (Xu et al., 2023), CLIP Score (Hessel et al., 2021), peak signal-to-noise ratio (PSNR), structural similarity (SSIM) (Wang et al., 2004), and learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018); for Wan2.1-1.3B, we use all 946 VBench (Huang et al., 2024) prompts and report the VBench score, PSNR, SSIM, and LPIPS. PSNR, SSIM, and LPIPS measure paired drift from full computation; IR, CLIP Score, and VBench are reference-free. Efficiency is measured by latency and floating-point operations (FLOPs). Selected FLUX.1-dev points are also evaluated with FID and kernel inception distance (KID) on COCO captions (Lin et al., 2014) and over four DrawBench seeds (Appendix E.2).

Implementation Details. All experiments are conducted on NVIDIA A100-80GB GPUs using PyTorch. DiT-XL/2 uses classifier-free guidance (CFG) at scale 1.5 and generates batches of 32 samples (tensor batch 64) that share one uncertainty score; FLUX.1-dev and Wan2.1-1.3B generate one prompt at a time. For each fixed backbone configuration, operating points are obtained by sweeping the trust threshold λ and labeled by relative compute (FLOPs of the denoising network relative to

Table 1: Main results on DiT-XL/2 for ImageNet 256×256 class-conditional generation.

Method DiT-XL/2	Speed				Quality				
	Latency (ms) ↓	Latency Speedup ↑	FLOPs (T) ↓	FLOPs Speedup ↑	FID ↓	sFID ↓	Inception Score ↑	Precision ↑	Recall ↑
Baseline (50 Steps)	358.44	1.00×	23.74	1.00×	2.282	4.290	237.53	0.8068	0.5959
Step Reduction (25 Steps)	185.94	1.93×	11.87	2.00×	2.892	4.583	231.24	0.7959	0.5924
Step Reduction (20 Steps)	151.88	2.36×	9.49	2.50×	3.581	4.941	222.11	0.7818	0.5761
L2C (NFE = 50)	309.06	1.16×	18.10	1.31×	2.265	**4.233**	243.65	0.8133	0.5887
Δ-DiT ($N = 3$)	243.13	1.47×	16.61	1.43×	3.440	4.756	209.22	0.7721	**0.6088**
FORA ($N = 2$)	219.06	1.64×	12.35	1.92×	2.649	4.555	234.58	0.8070	0.5851
ToCa ($N = 3$)	215.63	1.66×	10.23	2.32×	2.869	4.722	232.77	0.8020	0.5856
DuCa ($N = 3$)	201.25	1.78×	9.97	2.38×	2.856	4.625	231.97	0.8018	0.5795
TaylorSeer ($N = 4, O = 1$)	211.56	1.69×	7.61	3.12×	2.581	4.563	231.58	0.8020	0.5906
***KalmanCache+* (48% compute)**	212.19	1.69×	11.39	2.08×	**2.245**	4.287	**263.61**	**0.8246**	0.5797
FORA ($N = 3$)	168.13	2.13×	8.56	2.77×	3.253	5.382	228.36	0.7970	0.5653
ToCa ($N = 5$)	172.19	2.08×	7.44	3.19×	6.464	7.185	197.48	0.7421	0.5256
DuCa ($N = 5$)	151.25	2.37×	6.74	3.52×	6.161	6.759	197.53	0.7451	0.5229
TaylorSeer ($N = 6, O = 1$)	189.38	1.89×	5.71	4.15×	3.200	5.374	223.05	0.7904	0.5805
***KalmanCache+* (42% compute)**	187.19	1.91×	9.97	2.38×	**2.241**	**4.399**	**261.15**	**0.8211**	0.5786
***KalmanCache+* (38% compute)**	171.56	2.09×	9.02	2.63×	2.319	4.555	256.00	0.8149	**0.5816**
L2C (NFE = 20)	147.19	2.44×	7.39	3.21×	3.531	4.702	225.60	0.7890	0.5616
FORA ($N = 4$)	139.69	2.57×	6.66	3.56×	4.144	6.677	218.53	0.7857	0.5467
ToCa ($N = 7$)	155.31	2.31×	6.25	3.80×	8.610	8.767	179.14	0.7124	0.5223
DuCa ($N = 6$)	143.75	2.49×	6.29	3.78×	6.450	6.736	186.51	0.7301	0.5447
***KalmanCache+* (32% compute)**	150.94	2.37×	7.60	3.12×	**2.489**	**4.568**	**252.38**	**0.8108**	**0.5775**

full computation, labeled Baseline in the tables). The λ of each point (Table 9 in Appendix D.1), the fixed filter constants, the warmup length, seeds, and the latency and FLOPs protocols are in Appendix B.3. A rule for new deployments that sets λ from gate and compute logs alone met five predeclared compute targets within 2% (Appendix D). Relative to plain reuse on the same DiT-XL/2 FULL/SKIP schedule, the filter and residual memory add 2.12% sampling-loop time, included in all reported latencies, and under 0.3% of device memory (Appendix E.5).

4.2 Main Results

Class-conditional generation on DiT-XL/2. Table 1 groups methods into three latency regimes, as FLOPs savings need not translate into latency (TaylorSeer, N=4, O=1: 3.12× in FLOPs, 1.69× in latency). At 1.69×, *KalmanCache+* (48% compute) matches the baseline FID (2.245 vs. 2.282) and improves IS and Precision. In the faster regimes, the 42%-, 38%-, and 32%-compute points (1.91×, 2.09×, 2.37×) reach FID 2.241 (on par with the baseline), 2.319, and 2.489, vs. 3.200–8.610 for other caching methods. These results demonstrate that *KalmanCache+* accelerates DiT-XL/2 by up to 2.37× while leading the compared caching methods in FID, IS, and Precision in every regime, and in all five metrics at 2.37×.

Text-to-image generation on FLUX.1-dev. In Table 2, *KalmanCache+* keeps SSIM at 0.92, 0.88, and ∼0.8 at 1.99×, 2.34×, and 3.10–3.44×, whereas no other method exceeds 0.77 and those beyond 3.4× drop to the 0.6 range (vs. each method's own full computation). Its LPIPS at 3.10× is 0.199 vs. 0.368 for 17 steps at 2.88×, with ImageReward and CLIP Score near the baseline. This fidelity is consistent with its closed loop: it skips only at low predicted uncertainty and corrects later SKIP steps with FULL-step mismatches. These results demonstrate that on images *KalmanCache+* gives the lowest FID on DiT-XL/2 and the best PSNR, SSIM, and LPIPS on FLUX.1-dev among the compared caching methods in each reported latency regime.

Text-to-video generation on Wan2.1-1.3B. In Table 3, *KalmanCache+* (46% compute) is faster than TeaCache (l=0.1; 2.13× vs. 2.05×) while improving PSNR (29.55 vs. 23.16 dB), SSIM, and LPIPS. At the 2.00× FLOPs speedup of MagCache (K=2), the 50%-compute point (warmup W=18 instead of 15) improves all three (PSNR 31.20 vs. 28.53 dB); the 42%-compute point improves PSNR and SSIM over MagCache (K=4) at similar LPIPS and speed. TeaCache and MagCache use offline-calibrated curves, whereas *KalmanCache+* is corrected online. With VBench scores within 0.6 points of the caching method each point is compared with, the gain on Wan2.1-1.3B lies in fidelity to full computation. Figure 3 shows this fidelity on every backbone: *KalmanCache+* keeps the layout and content of full computation, whereas several compared caching methods alter the scene. Overall,

Table 2: Main results on FLUX.1-dev for text-to-image generation.

Method FLUX.1-dev	Speed				Quality				
	Latency (s) ↓	Latency Speedup ↑	FLOPs (T) ↓	FLOPs Speedup ↑	Image Reward ↑	CLIP Score ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline (50 Steps)	24.50	1.00×	3719.50	1.00×	1.029	33.24	∞	1.0000	0.0000
Step Reduction (25 Steps)	12.38	1.98×	1859.75	2.00×	0.958	33.24	18.68	0.7645	0.2707
Step Reduction (20 Steps)	9.96	2.46×	1487.80	2.50×	0.967	33.08	17.13	0.7251	0.3261
Step Reduction (17 Steps)	8.51	2.88×	1264.63	2.94×	0.923	32.67	16.36	0.6968	0.3676
Δ-DiT ($N = 3$)	13.70	1.79×	2083.36	1.79×	0.938	32.81	14.86	0.6414	0.4381
ToCa ($N = 5$)	14.11	1.74×	1125.11	3.31×	1.022	32.76	17.49	0.6901	0.3446
FORA ($N = 2$)	12.20	2.01×	1860.32	2.00×	**1.052**	32.85	17.12	0.7099	0.3262
***KalmanCache+* (43% compute)**	12.29	1.99×	1607.29	2.31×	1.027	**33.31**	**28.07**	**0.9193**	**0.0748**
TeaCache ($l = 0.4$)	10.52	2.33×	1488.29	2.50×	1.013	**33.43**	18.60	0.7550	0.2797
TaylorSeer ($N = 3, O = 1$)	9.69	2.53×	1339.75	2.78×	**1.061**	32.86	18.92	0.7608	0.2467
DiCache ($\delta = 0.2$)	9.60	2.55×	1381.26	2.69×	0.929	32.73	16.28	0.6870	0.3875
FORA ($N = 3$)	8.63	2.84×	1265.38	2.94×	1.005	32.68	15.28	0.6447	0.4125
ToCa ($N = 8$)	10.89	2.25×	852.19	4.36×	0.955	32.58	16.20	0.6050	0.4716
***KalmanCache+* (36% compute)**	10.46	2.34×	1332.10	2.79×	1.036	33.26	**25.41**	**0.8834**	**0.1117**
***KalmanCache+* (28% compute)**	8.71	2.81×	1042.05	3.57×	1.035	33.22	22.23	0.8271	0.1782
TaylorSeer ($N = 5, O = 1$)	7.10	3.45×	893.54	4.16×	**1.026**	32.85	15.98	0.6677	0.3816
TeaCache ($l = 0.8$)	6.87	3.56×	893.30	4.16×	0.873	32.57	16.41	0.6823	0.3891
FORA ($N = 4$)	6.84	3.58×	967.91	3.84×	0.990	32.66	14.41	0.6099	0.4669
***KalmanCache+* (26% compute)**	7.89	3.10×	975.11	3.81×	1.022	**33.15**	**21.54**	**0.8082**	**0.1991**
***KalmanCache+* (24% compute)**	7.13	3.44×	893.30	4.16×	1.016	33.14	20.82	0.7870	0.2293

Table 3: Main results on Wan2.1-1.3B for text-to-video generation.

Method Wan2.1-1.3B	Speed				Quality			
	Latency (s) ↓	Latency Speedup ↑	FLOPs (P) ↓	FLOPs Speedup ↑	PSNR ↑	SSIM ↑	LPIPS ↓	VBench Score (%) ↑
Baseline (50 Steps)	192.08	1.00×	28.506	1.00×	∞	1.0000	0.0000	80.95
Step Reduction (20 Steps)	78.91	2.43×	11.402	2.50×	15.00	0.5888	0.4034	80.05
TeaCache ($l = 0.1$)	93.59	2.05×	13.684	2.08×	23.16	0.8263	0.1137	**80.54**
***KalmanCache+* (46% compute)**	90.34	2.13×	13.000	2.19×	**29.55**	**0.9119**	**0.0574**	80.07
MagCache ($K = 2$)	97.32	1.97×	14.254	2.00×	28.53	0.9142	0.0476	**80.79**
***KalmanCache+* (50% compute)**	98.07	1.96×	14.254	2.00×	**31.20**	**0.9313**	**0.0449**	80.40
MagCache ($K = 4$)	83.71	2.29×	11.974	2.38×	26.35	0.8834	0.0715	**80.84**
***KalmanCache+* (42% compute)**	85.42	2.25×	12.088	2.36×	**27.92**	**0.8908**	**0.0712**	80.29

these results demonstrate that *KalmanCache+* improves the speed–fidelity trade-off of caching not only on images but also on video, across all three generation tasks.

4.3 Ablation Studies

Table 4 isolates the FULL/SKIP decision rule and the SKIP reconstruction at approximately matched budgets. The † run replays the gate's schedule for the first DiT-XL/2 batch or FLUX.1-dev prompt on all samples; reconstruction variants use this schedule, and decision variants the closed-loop reconstruction with the gate's λ (w_i capped at one), so decision rows differ only in their schedules.

Effectiveness of the trust gate in FULL/SKIP decisions. We compare the gate with two reliability-blind rules: a uniform interval and an input-change-derived schedule (TeaCache's signal without its rescaling). The two rules swap ranks across backbones (the input-change-derived schedule beats the uniform interval on DiT-XL/2, 2.295 vs. 2.454 FID-50K, but trails it on FLUX.1-dev, 16.49 vs. 19.28 dB PSNR), whereas the gate's fixed schedule (†) beats both on all four metrics, and the adaptive gate improves further. This ranking also holds at a second seed; on FLUX.1-dev at seed 0, the † schedule also beats the uniform interval when both use plain reuse, and the adaptive gate beats TeaCache's rescaled rule (Appendix D.3). These results demonstrate that placing FULL steps where the predicted uncertainty is high outperforms the tested reliability-blind rules in DiT-XL/2 FID-50K and FLUX.1-dev fidelity at these budgets.

Effect of closed-loop feedback on SKIP reconstruction. On the fixed schedule (†), plain reuse gives the worst FID-50K (2.953); extrapolation alone lowers it to 2.325, and the closed-loop reconstruction to 2.251 while also beating both in FLUX.1-dev PSNR, SSIM, and LPIPS. Unlike these open-

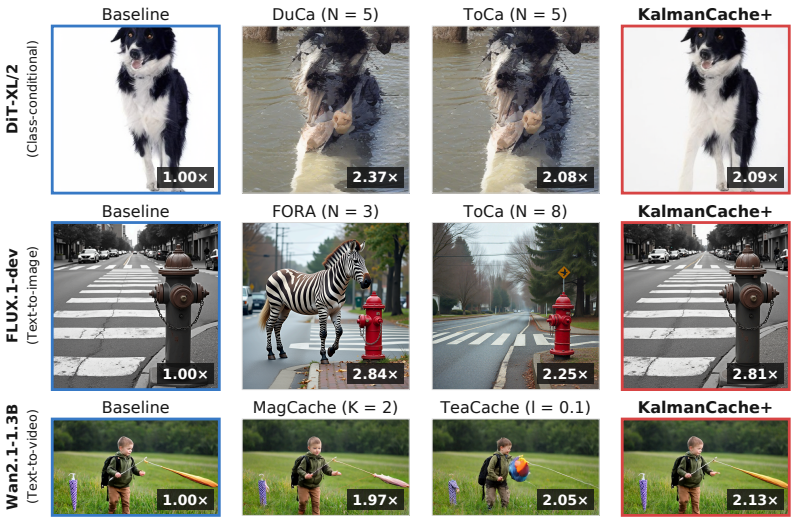


Figure 3: Qualitative comparison with the compared caching methods on DiT-XL/2 (top), FLUX.1-dev (middle), and Wan2.1-1.3B (bottom); numbers are latency speedups.

Table 4: Ablation of the FULL/SKIP decision rule and the SKIP reconstruction at approximately matched budgets (21 of 50 FULL steps on DiT-XL/2, 13 of 50 on FLUX.1-dev; seed 0). On FLUX.1-dev, each non-† row differs from the † row in all three metrics (paired 95% bootstrap CIs exclude zero). †Same run in both blocks.

		DiT-XL/2	FLUX.1-dev		
Component	Variant	FID-50K ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FULL/SKIP decision	Uniform interval	2.454	19.28	0.749	0.273
	Input-change-derived schedule	2.295	16.49	0.608	0.455
	Trust gate's schedule, fixed†	2.251	20.85	0.794	0.216
	Trust gate, adaptive (*KalmanCache+*)	**2.241**	**21.54**	**0.808**	**0.199**
SKIP reconstruction	Reuse last residual	2.953	20.37	0.777	0.248
	+ Residual extrapolation	2.325	19.99	0.785	0.227
	+ Closed-loop reconstruction†	**2.251**	**20.85**	**0.794**	**0.216**

loop variants, it corrects the extrapolation with the mismatch observed at the preceding FULL step and weights it by the filter's uncertainty. These results show that the closed-loop reconstruction improves FID-50K and paired fidelity over the two tested open-loop variants. Together, the two blocks of Table 4 show that both components contribute at approximately matched budgets: with the reconstruction fixed, the trust gate improves on the reliability-blind rules, and with the schedule fixed, the closed-loop reconstruction improves on the open-loop variants.

5 Conclusion

This paper presents *KalmanCache+*, which casts training-free diffusion caching as sequential residual estimation in a Predict–Trust–Correct cycle. Its trust gate skips a step only while the uncertainty score of a channel-wise Kalman filter stays below the trust threshold, and each post-warmup FULL step is fed back to the filter (channel-wise feedback) and to the residual memory (tensor-level feedback, C-DOB). Experimental results show that *KalmanCache+* improves the speed–fidelity trade-off across class-conditional, text-to-image, and text-to-video generation, with latency speedups of 1.91×, 3.10×, and 2.13× on DiT-XL/2, FLUX.1-dev, and Wan2.1-1.3B; notably, on the image backbones at approximately matched budgets, its trust gate beats two reliability-blind rules and its closed-loop reconstruction two open-loop ones. These results highlight *KalmanCache+* as a practical, training-free way to accelerate diffusion transformers by turning FULL steps into observations that inform when and how later steps are skipped. Its main limitations are an unamortized warmup on short samplers, a gate shared within each DiT-XL/2 batch, and a trust threshold set per deployment (Appendix F).

Reproducibility Statement

Appendices A and B specify the estimator, the inference procedure, and the per-model settings, including the evaluation protocol of each backbone (Table 7); Table 9 lists the trust threshold λ behind every main-table operating point, and Appendix D.1 gives the rule for setting it on a new deployment. The implementation will be released upon acceptance.

AI Use Statement

Large language model (LLM) assistants were used to aid and polish the writing and for retrieval and discovery of related work. The authors take full responsibility for the content of this paper.

References

Black Forest Labs. FLUX.1 [dev]. Model card, 2024. URL https://huggingface.co/black-forest-labs/FLUX.1-dev.

Daniel Bolya and Judy Hoffman. Token merging for fast Stable Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4599–4603, 2023.

Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. DiCache: Let diffusion model determine its own cache. In *International Conference on Learning Representations*, 2026.

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Δ-DiT: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024.

Dahun Choi and Hyun Kim. GradQ-ViT: Robust and efficient gradient quantization for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16019–16027, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, pp. 12606–12633, 2024.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in Neural Information Processing Systems*, 36:16716–16728, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, 2021.

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. *Advances in Neural Information Processing Systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Rudolf Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Nam Joon Kim and Hyun Kim. Trunk pruning: Highly compatible channel pruning for convolutional neural networks without fine-tuning. *IEEE Transactions on Multimedia*, 26:5588–5599, 2024.

Nam Joon Kim, Jongho Lee, and Hyun Kim. HyQ: Hardware-friendly post-training quantization for CNN-transformer hybrid networks. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4291–4299, 2024.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

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*, pp. 740–755, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025a.

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

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

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. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *Advances in Neural Information Processing Systems*, 37: 133282–133304, 2024a.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15762–15772, 2024b.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. MagCache: Fast video generation with magnitude-aware cache. *Advances in Neural Information Processing Systems*, 38: 34348–34380, 2025.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. In *International Conference on Machine Learning*, pp. 7958–7968, 2021.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Andrew P. Sage and Gary W. Husa. Adaptive filtering with unknown prior statistics. In *Proceedings of the Joint Automatic Control Conference*, pp. 760–769, 1969.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

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

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, 29, 2016.

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

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252, 2023.

Sebastian Trimpe and Raffaello D'Andrea. Event-based state estimation with variance-based triggering. *IEEE Transactions on Automatic Control*, 59(12):3266–3281, 2014.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Junyi Wu, Haoxuan Wang, Yuzhang Shang, Mubarak Shah, and Yan Yan. PTQ4DiT: Post-training quantization for diffusion transformers. *Advances in Neural Information Processing Systems*, 37: 62732–62755, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:49842–49869, 2023.

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*, 2025.

Chang Zou, Shikang Zheng, Evelyn Zhang, Runlin Guo, Haohang Xu, Zhengyi Shi, Conghui He, Xuming Hu, and Linfeng Zhang. Rethinking token-wise feature caching: Accelerating diffusion transformers with dual feature caching. *IEEE Transactions on Image Processing*, 35:6211–6220, 2026.

A Estimator Details

Appendices A.1–A.5 specify the channel-wise filter and the tensor-level SKIP reconstruction in the notation of Section 3; Appendix A.6 describes what differs on Wan2.1-1.3B. The overview, the cycle summary of Section 3, and Figure 2 omit the module index (m) and shorten subscripts; the figure writes $\mathcal{R}, \mathcal{E}$ as R, E. Table 5 collects the notation in one place.

Table 5: Notation of *KalmanCache+*; the module index (m) and the channel index $c = 1, \ldots, C$ are omitted.

Symbol	Meaning
i, T, W	sampling step; number of steps (50); warmup length (main results: 4 on the image backbones; 15 on Wan2.1-1.3B, 18 at its 50%-compute point)
$\mathcal{P}$	protected steps $\{1, \ldots, W\} \cup \{T\}$, always FULL
j, k, Δ_i	most recent and second most recent FULL steps before step i; gap $\Delta_i = i - j$
$z_i, y_i, \mathcal{R}_i$	input and full-computation output of the cached block stack $B_1, \ldots, B_L$; residual $\mathcal{R}_i = y_i - z_i$
$\widehat{\mathcal{R}}_i, \widehat{y}_i, e_i$	SKIP reconstruction of the residual; SKIP output $\widehat{y}_i = z_i + \widehat{\mathcal{R}}_i$; residual approximation error $e_i = \mathcal{R}_i - \widehat{\mathcal{R}}_i$
o_i, Ω	channel means of $\mathcal{R}_i$, the filter's only measurement, taken over the non-channel indices Ω
$s = [\mu, \nu, b]^\top$	per-channel state: level (channel mean), trend per step, and persistent bias
P, P_{ref}	3×3 error covariance of s; reference covariance $P_{\text{ref}} = P_W$ at the end of warmup
$\widehat{s}_{i\|i-1}, \widehat{P}_{i\|i-1}$	prediction for step i from the FULL checkpoint $(s_{j\|j}, P_{j\|j})$, the estimate after the update at step j; written $(\widehat{s}_i, \widehat{P}_i)$ and (s_j, P_j) in the overview, the cycle summary, and Figure 2
A_Δ, Q_Δ	state transition and process noise over a gap of Δ steps (Appendix A.1)
q, q_b, V	adapted process-noise scale; fixed bias process-noise scale (q_b=0.01); adapted measurement-noise scale
ι, G, H	innovation (observed minus predicted level); Kalman gain; observation row $H = [1, 0, 0]$
$\alpha_i, \alpha_{\text{floor}}, N_i$	Sage–Husa rate $\max(\alpha_{\text{floor}}, 2/(N_i + 1))$; its floor (0.05); count of the warmup observations and earlier post-warmup updates
u_i, ε	uncertainty score (channel-averaged predicted covariance trace over that of P_{ref}, Eq. 4); small constant for numerical stability (the implementation uses ε=0 and instead forces FULL when the channel-averaged trace of P_{ref} is below 10^{-12}; Algorithm 1)
$\lambda, \lambda_{\text{eff}}, d_i$	trust threshold; threshold the gate compares u_i with (λ on the image backbones, $\lambda\rho_j$ on video); FULL/SKIP decision
w_i	SKIP blend weight $\min(1, u_i/\lambda)$, equal to u_i/λ at every SKIP step that the trust gate selects (Appendix A.5)
$\mathcal{E}_i^{\text{lin}}, \mathcal{E}_i^{\text{vid}}, \mathcal{E}_i$	linear extrapolation from the last two FULL residuals; its damped form for video; the extrapolation SKIP uses ($\mathcal{E}_i^{\text{lin}}$ on the image backbones, $\mathcal{E}_i^{\text{vid}}$ on video)
D_j	C-DOB mismatch $\mathcal{R}_j - \mathcal{E}_j^{\text{lin}}$ stored at FULL step j
$\rho, \alpha^\rho, n_\rho$	video autocorrelation estimate in $[0, 1]$; its update rate $\max(0.05, 2/(n_\rho + 1))$; number of estimates of the branch in the current video (Appendix A.6)

A.1 State Transition and Process Noise

The predict step (Eq. 3) propagates the checkpoint of the most recent FULL step j over the gap $\Delta_i = i - j$ with

$$A_{\Delta_i} = \begin{bmatrix} 1 & \Delta_i & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad Q^{(m)}_{\Delta_i,c} = q^{(m)}_{j,c} \begin{bmatrix} \Delta_i^3/3 & \Delta_i^2/2 & 0 \\ \Delta_i^2/2 & \Delta_i & 0 \\ 0 & 0 & q_b\,\Delta_i \end{bmatrix}, \tag{8}$$

where $q^{(m)}_{j,c}$ is the process-noise scale stored at step j (Appendix A.4) and q_b is the fixed bias process-noise scale (Appendix B.3). The gap counts sampling steps, not diffusion time, so ν and q are defined per step. A_{Δ_i} maps the checkpoint at step j directly to step i, so the bias b is added to the level once per prediction, whatever the gap.

A.2 State Initialization and Reference Covariance

The warmup steps $1, \ldots, W$ run FULL. Let $a^{(m)}_{1,c}, \ldots, a^{(m)}_{n,c}$ be the channel-mean observations used for initialization: $a^{(m)}_{t,c} = o^{(m)}_{t,c}$ (Eq. 6) and $n = W$ for DiT-XL/2 and FLUX.1-dev (Wan2.1-1.3B: Appendix A.6). With first differences $\delta^{(m)}_{t,c} = a^{(m)}_{t,c} - a^{(m)}_{t-1,c}$ and second differences $\gamma^{(m)}_{t,c} =$

$\delta_{t,c}^{(m)} - \delta_{t-1,c}^{(m)}$, each channel is initialized at the last warmup step as

$$\mu_{W,c}^{(m)} = a_{n,c}^{(m)}, \qquad \nu_{W,c}^{(m)} = \delta_{n,c}^{(m)}, \qquad b_{W,c}^{(m)} = 0, \tag{9}$$

$$P_{W,c}^{(m)} = \mathrm{diag}\Big(\mathrm{Var}_t\big(a_{t,c}^{(m)}\big),\, \mathrm{Var}_t\big(\delta_{t,c}^{(m)}\big),\, \mathrm{Var}_t\big(\gamma_{t,c}^{(m)}\big)\Big), \tag{10}$$

$$q_{W,c}^{(m)} = \max\Big(\tfrac{1}{n-2}\textstyle\sum_{t=3}^{n}\big(\gamma_{t,c}^{(m)}\big)^2,\, 10^{-10}\Big), \qquad V_{W,c}^{(m)} = \max\Big(\mathrm{Var}_\Omega\big(\mathcal{R}_{W,c}^{(m)}\big)/|\Omega|,\, 10^{-12}\Big), \tag{11}$$

with unbiased sample variances: q is the mean squared second difference, and V is the squared standard error of the last warmup channel mean (the variance of $\mathcal{R}_{W,c}^{(m)}$ over Ω divided by $|\Omega|$). These values form the first FULL checkpoint ($j = W$), and $P_{W,c}^{(m)}$ is also the reference covariance $P_{\mathrm{ref},c}^{(m)}$ of Eq. 4.

A.3 Channel-wise Feedback: Measurement Update

At a FULL step, the filter observes the level through $H = [1, 0, 0]$, and the channel-wise innovation is $\iota_{i,c}^{(m)} = o_{i,c}^{(m)} - H\,\widehat{s}_{i|i-1,c}^{(m)}$ with $o_{i,c}^{(m)}$ from Eq. 6. The gain, corrected state, and Joseph-form covariance use the measurement-noise scale $V_{i,c}^{(m)}$, which is updated from $\iota_{i,c}^{(m)}$ just before the gain (Appendix A.4):

$$G_{i,c}^{(m)} = \widehat{P}_{i|i-1,c}^{(m)} H^\top \Big(H\,\widehat{P}_{i|i-1,c}^{(m)} H^\top + V_{i,c}^{(m)}\Big)^{-1}, \qquad s_{i|i,c}^{(m)} = \widehat{s}_{i|i-1,c}^{(m)} + G_{i,c}^{(m)}\,\iota_{i,c}^{(m)}, \tag{12}$$

$$P_{i|i,c}^{(m)} = \Big(I - G_{i,c}^{(m)} H\Big)\,\widehat{P}_{i|i-1,c}^{(m)}\,\Big(I - G_{i,c}^{(m)} H\Big)^\top + G_{i,c}^{(m)}\,V_{i,c}^{(m)}\,G_{i,c}^{(m)\top}. \tag{13}$$

A.4 Sage–Husa Adaptive Noise Estimation

At every FULL step after warmup, the noise scales V and q are re-estimated from the innovation (Sage & Husa, 1969) with the rate

$$\alpha_i = \max\Big(\alpha_{\mathrm{floor}},\, \tfrac{2}{N_i+1}\Big), \tag{14}$$

where α_{floor} is a fixed floor (Appendix B.3) and N_i counts the warmup observations plus the earlier post-warmup FULL updates (one counter per filter). Before the gain, V is updated as

$$\tilde{V}_{i,c}^{(m)} = \max\Big(\big(\iota_{i,c}^{(m)}\big)^2 - \big[\widehat{P}_{i|i-1,c}^{(m)}\big]_{11},\, 10^{-12}\Big), \tag{15}$$

$$V_{i,c}^{(m)} = \max\Big((1-\alpha_i)\,V_{j,c}^{(m)} + \alpha_i\,\tilde{V}_{i,c}^{(m)},\, 10^{-12}\Big). \tag{16}$$

After the correction, q is updated from the $(1,1)$ element of the Sage–Husa process-noise estimate Ξ, divided by the $(1,1)$ element $\Delta_i^3/3$ of $Q_{\Delta_i,c}^{(m)}/q_{j,c}^{(m)}$ to normalize for the gap:

$$\Xi_{i,c}^{(m)} = G_{i,c}^{(m)}\iota_{i,c}^{(m)}\big(G_{i,c}^{(m)}\iota_{i,c}^{(m)}\big)^\top + P_{i|i,c}^{(m)} - A_{\Delta_i}P_{j|j,c}^{(m)}A_{\Delta_i}^\top, \tag{17}$$

$$\tilde{q}_{i,c}^{(m)} = \max\Big(\max\big(\big[\Xi_{i,c}^{(m)}\big]_{11},\, 10^{-12}\big)\big/\big(\Delta_i^3/3\big),\, 10^{-12}\Big), \tag{18}$$

$$q_{i,c}^{(m)} = \max\Big((1-\alpha_i)\,q_{j,c}^{(m)} + \alpha_i\,\tilde{q}_{i,c}^{(m)},\, 10^{-10}\Big). \tag{19}$$

A.5 Tensor-Level Feedback (C-DOB) and SKIP Reconstruction

For the image backbones, a SKIP step reconstructs the residual by Eq. 5 with $\mathcal{E}_i^{(m)} = \mathcal{E}_i^{\mathrm{lin}\,(m)}$, extrapolated from the two most recent FULL steps $k < j$. The mismatch stored by the C-DOB is $D_j^{(m)} = \mathcal{R}_j^{(m)} - \mathcal{E}_j^{\mathrm{lin}\,(m)}$ (Eq. 7 at step j), where $\mathcal{E}_j^{\mathrm{lin}\,(m)}$ is extrapolated to step j from the two FULL steps before j. It is first computed at the third FULL step, inside warmup ($W \geq 3$), and refreshed at every later FULL step, so every SKIP step has a memory term. The blend weight $w_i^{(m)}$ scales the correction by the uncertainty score and is not an inverse-variance fusion weight; the implementation clips it, $w_i^{(m)} = \min(1, u_i^{(m)}/\lambda)$, which equals $u_i^{(m)}/\lambda$ at every SKIP step that the trust gate selects, because $u_i^{(m)} < \lambda$ there; a SKIP step forced by another rule (the decision variants of Appendix D.3) can have $u_i^{(m)} \geq \lambda$, and the clip then sets $w_i^{(m)}$=1.

A.6 Video Backbone: Guidance Branches and Autocorrelation Damping

Wan2.1-1.3B keeps a separate filter and cache for each guidance branch (Appendix B.2). The branches pool their warmup channel means in execution order (conditional before unconditional at each step), so the conditional filter initializes from the first $n{=}2W{-}1$ pooled observations and the unconditional filter from all $n{=}2W$; their counters N_i start at these values, and each branch takes V from its own last warmup residual.

Each branch also estimates $\rho \in [0, 1]$, a clipped, uncentered regression coefficient between the channel-mean vectors $o_i^{(m)} = (o_{i,1}^{(m)}, \ldots, o_{i,C}^{(m)})$ of consecutive FULL steps, used as an autocorrelation surrogate. At every FULL step i with a previous FULL step j in the same branch (warmup included),

$$\rho_{\text{new}} = \text{clip}\Big(\langle o_i^{(m)}, o_j^{(m)}\rangle \big/ \max\big(\langle o_j^{(m)}, o_j^{(m)}\rangle,\, 10^{-12}\big),\, 0,\, 1\Big), \tag{20}$$

$$\rho_i = (1 - \alpha^\rho)\,\rho_j + \alpha^\rho \rho_{\text{new}}, \qquad \alpha^\rho = \max\big(0.05,\, 2/(n_\rho + 1)\big), \tag{21}$$

where n_ρ counts the branch's estimates in the current video, including the current one, so $\alpha^\rho = 1$ and $\rho_i = \rho_{\text{new}}$ at the first estimate. At an unprotected step i, the gate and the damped extrapolation below use ρ_j, the value stored at the last FULL step j (before the current step's observation). The gate selects FULL when $u_i^{(m)} \geq \lambda\rho_j$, so the reported Wan2.1-1.3B λ values are nominal (the effective threshold $\lambda\rho_j$ never exceeds λ). The SKIP blend of Eq. 5 uses $\mathcal{E}_i^{(m)} = \mathcal{E}_i^{\text{vid}\,(m)}$, the damped extrapolation

$$\mathcal{E}_i^{\text{vid}\,(m)} = \mathcal{R}_j^{(m)} + \rho_j^{\Delta_i}\big(\mathcal{E}_i^{\text{lin}\,(m)} - \mathcal{R}_j^{(m)}\big), \tag{22}$$

while the blend weight keeps λ, $w_i^{(m)} = \min(1, u_i^{(m)}/\lambda)$, and the stored mismatch keeps the undamped $\mathcal{E}^{\text{lin}}$.

B Inference Algorithm and Implementation

This appendix gives the inference procedure of *KalmanCache+* (Algorithm 1), the cacheable module (Appendix B.2), and the per-backbone settings and evaluation protocol (Appendix B.3).

B.1 Inference Algorithm

Algorithm 1 and Table 6 describe one filter; for Wan2.1-1.3B it runs once per guidance branch at every denoising step, conditional branch first, with the pooled warmup of Appendix A.6. At a FULL step i, the stored mismatch uses $\mathcal{R}_k$ and $\mathcal{R}_j$, and, after warmup, the filter update uses the FULL checkpoint of step j with $\Delta_i = i - j$, all taken before the residual memory is shifted. What persists between steps is fixed in size: per channel a three-dimensional state, its 3×3 covariance, two scalar noise terms, and the warmup reference P_{ref} of the uncertainty score; scalar bookkeeping; and the residual memory (the last two FULL residuals and the stored mismatch D), plus ρ for video. SKIP steps read the filter without modifying it, and every FULL step after warmup, including the final one, corrects it.

B.2 Cacheable Module

The cacheable module m is the transformer-block stack in hidden (token) space: $z_i^{(m)}$ is the hidden token sequence entering the first block, after the backbone's input embedding or projection; $y_i^{(m)}$ is the hidden state leaving the last block, before the final layer; and $\mathcal{R}_i^{(m)} = y_i^{(m)} - z_i^{(m)}$ (Eq. 1). Only the block stack is skipped: the embeddings, the conditioning, the final projection and unpatchify, the guidance combination, the sampler update, and the variational autoencoder (VAE) are always executed. The cached tensors are $[64, 256, 1152]$ for DiT-XL/2 (tensor batch 64 for 32 samples and two guidance branches, 256 tokens, 28 blocks), $[1, 4096, 3072]$ for FLUX.1-dev (the image-token stream after its 19 dual-stream and 38 single-stream blocks; the text stream is not cached, and FLUX.1-dev uses guidance embedding rather than a guidance batch), and $[1, 32760, 1536]$ per guidance branch for Wan2.1-1.3B (30 blocks). The channel index c is the hidden-dimension axis, and Ω in Eq. 6 averages over the batch (including both guidance branches for DiT-XL/2) and all token axes. DiT-XL/2 and FLUX.1-dev make one FULL/SKIP decision per denoising step, whereas

Algorithm 1 Inference with *KalmanCache+*

Require: Trust threshold λ, warmup length W, number of steps T, protected-step set $\mathcal{P} = \{1, \dots, W\} \cup \{T\}$ (one filter; indices (m) and c omitted; $k < j$ are the two most recent FULL steps); $\lambda_{\text{eff}} = \lambda$ for the image backbones and $\lambda\rho_j$ for video, with ρ_j from the last FULL step (App. A.6)

for $i = 1, \dots, T$ **do**
  if $i > W$ **then**
    Predict: $\widehat{s}_{i|i-1} = A_{\Delta_i} s_{j|j}$, $\widehat{P}_{i|i-1} = A_{\Delta_i} P_{j|j} A_{\Delta_i}^\top + Q_{\Delta_i}$, $\Delta_i = i - j$ (Eq. 3; App. A.1)
  end if
  if $i \notin \mathcal{P}$ **then**
    Compute u_i via Eq. 4 (FULL if the denominator is below 10^{-12} or u_i is not finite)
    if $u_i < \lambda_{\text{eff}}$ **then**
      $w_i = u_i/\lambda$; $\widehat{\mathcal{R}}_i = (1 - w_i)\,\mathcal{R}_j + w_i(\mathcal{E}_i + D_j)$ with $\mathcal{E}_i = \mathcal{E}_i^{\text{lin}}$ (Eq. 5, App. A.5; video: $\mathcal{E}_i^{\text{vid}}$, App. A.6); $\widehat{y}_i = z_i + \widehat{\mathcal{R}}_i$; **continue** ▷ SKIP: reads the filter, modifies nothing
    end if
  end if
  Run FULL: $\mathcal{R}_i = y_i - z_i$, $o_i = \text{Mean}_\Omega(\mathcal{R}_i)$ (Eq. 6)
  If two earlier FULL residuals exist: $D_i = \mathcal{R}_i - \mathcal{E}_i^{\text{lin}}$, $\mathcal{E}_i^{\text{lin}} = \mathcal{R}_j + \frac{i-j}{j-k}(\mathcal{R}_j - \mathcal{R}_k)$ (Eqs. 5, 7)
  Update ρ (video only, if one earlier FULL observation exists), then the residual memory
  if $i \le W$ **then**
    Record o_i; at $i = W$, initialize s, P, q, V, and P_{ref} (App. A.2)
  else
    $\iota_i = o_i - H\widehat{s}_{i|i-1}$ with $H = [1, 0, 0]$; $\alpha_i = \max(\alpha_{\text{floor}}, 2/(N_i + 1))$ (App. A.4)
    $V_i = \max\left((1 - \alpha_i)V_j + \alpha_i \max(\iota_i^2 - [\widehat{P}_{i|i-1}]_{11}, 10^{-12}),\ 10^{-12}\right)$
    $G_i = \widehat{P}_{i|i-1} H^\top / ([\widehat{P}_{i|i-1}]_{11} + V_i)$; $s_{i|i} = \widehat{s}_{i|i-1} + G_i \iota_i$
    $P_{i|i} = (I - G_i H)\widehat{P}_{i|i-1}(I - G_i H)^\top + G_i V_i G_i^\top$
    $q_i = \max\left((1 - \alpha_i)q_j + \alpha_i \tilde{q}_i,\ 10^{-10}\right)$, $\tilde{q}_i$ from App. A.4 ▷ new FULL checkpoint
  end if
end for

Table 6: Operations of *KalmanCache+* at denoising step i; j is the most recent FULL step and $\Delta_i = i - j$. SKIP and FULL execute Trust's decision (Section 3.2); in Correct (Section 3.3), the FULL-row correction with Adapt is the channel-wise feedback and Memory the C-DOB.

Operation	What happens	Reference
Protected	steps $1, \dots, W$ and step T always run FULL, regardless of u_i	Sec. 3.2
Predict	state and covariance advance by Δ_i steps from the FULL checkpoint of step j	App. A.1
Trust	u_i: channel-averaged predicted covariance trace over its warmup reference; FULL is forced if the reference denominator is below 10^{-12} or u_i is not finite	Eq. 4
SKIP	unprotected step not forced to FULL, with $u_i < \lambda_{\text{eff}}$ (λ; $\lambda\rho$ for video): blend cached residual, extrapolation, and stored mismatch with $w_i = u_i/\lambda$; the filter is read but not modified	App. A.5, A.6
FULL	every step that is not SKIP: observe the residual; at step W, initialize the state, covariance, noise scales, and P_{ref} from the warmup residuals; after warmup, correct the state and update the covariance in Joseph form; video: update ρ once one earlier FULL observation exists	App. A.2, A.3, A.6
Adapt	every FULL step after warmup: update V before the gain and q after the correction, with rate $\alpha_i = \max(\alpha_{\text{floor}}, 2/(N_i + 1))$	App. A.4
Memory	every FULL step: once two earlier FULL residuals exist, stored mismatch $D_i = \mathcal{R}_i - \mathcal{E}_i^{\text{lin}}$; then shift the residual memory	Eq. 7, App. A.5

Wan2.1-1.3B keeps separate filters and caches for the conditional and unconditional branches and makes a separate decision for each.

B.3 PER-MODEL SETTINGS

Table 7 gives the evaluation protocol of each backbone, and Table 9 the λ of every operating point. FORA on FLUX.1-dev and Δ-DiT are run in the TaylorSeer codebase (Liu et al., 2025b), on FLUX.1-dev in its FORA and Δ-DiT modes with the step-type selection patched so that every SKIP step uses the FORA or Δ-DiT reuse rather than a ToCa step, and on DiT-XL/2 in a Δ-DiT mode we added; TeaCache and MagCache on Wan2.1-1.3B use their released forward functions and coefficients in a common runtime based on the Wan2.1 code of DiCache (Bu et al., 2026). The two fixed filter constants, the floor on the Sage–Husa adaptation rate and the bias process-noise scale, are $\alpha_{\text{floor}}{=}0.05$ and $q_b{=}0.01$ for all backbones and are not retuned (sensitivity: Appendix E.3). Latency is wall-clock time per sample on one A100; FLOPs are those of the denoising network over the executed FULL steps and exclude cache-side operations, which the latencies include (Appendix E.5). On FLUX.1-dev, the methods run in two implementations with full-computation latencies of 23.28 s and 25.71 s: TaylorSeer, FORA, ToCa, and Δ-DiT in the TaylorSeer codebase, and TeaCache, DiCache, and *KalmanCache+* in the diffusers library. The Baseline latency of Table 2 (24.50 s) is the mean of the two, each step-reduction latency is likewise the mean of its runs in the two implementations, and every FLUX.1-dev latency speedup, including those in Figure 3, is taken relative to this mean. Measured against the full computation of their own implementation instead, the latency speedups of the methods run in diffusers, including *KalmanCache+*, would each be about 5% higher, and those of the four methods run in the TaylorSeer codebase about 5% lower; as with the paired metrics (Appendix E.2), latency speedups are therefore strictly comparable only within one implementation.

Table 7: Evaluation protocol of the main results (Tables 1–3).

	DiT-XL/2	FLUX.1-dev	Wan2.1-1.3B
Inputs	50,000 samples, labels i.i.d. uniform over the 1,000 ImageNet classes	200 DrawBench prompts, one image each; selected points also over seeds 1–4 and on 5,000 COCO captions (Appendix E.2)	946 VBench prompts (944 distinct), one video each
Output	$256{\times}256$	$1024{\times}1024$	81 frames, $832{\times}480$
Sampler	DDIM, 50 steps	rectified flow (Euler), 50 steps	UniPC with shift 5, 50 steps
Guidance	CFG 1.5	guidance embedding 3.5	CFG 5
Seed	0; noise and labels drawn in sequence from one generator	0 for every prompt	2 for every prompt
Batch	32 (one uncertainty score per batch)	1	1
Warmup W	4	4	15 (18 at the 50% point)
Metrics	FID-50K, sFID, IS, Precision, Recall	IR, CLIP Score, PSNR, SSIM, LPIPS	VBench score, PSNR, SSIM, LPIPS
Reference	ImageNet reference batch (FID, sFID, Precision, Recall)	each method's own full computation, same prompt and seed	one full computation shared by all methods, same prompt and seed
Latency	batch-iteration time at batch 32 (sampling, VAE decoding, output handling) divided by 32	synchronized pipeline call per prompt, after three untimed runs	one video, including both guidance branches, VAE decoding, and video writing
FLOPs	analytic	forward hooks, 10 prompts	forward hooks, 5 prompts

Control sets. Several later analyses use two paired control sets: for DiT-XL/2, 100 class-conditional samples generated one at a time (per-image gating; classes drawn with a fixed generator, sample seeds 0–99, CFG 1.5), disjoint from the 20-image pilot of Appendix D.2; for Wan2.1-1.3B, 50 VBench prompts stratified across its 16 dimensions, at seed 2, on which FULL ratios count the conditional branch only and are therefore distinct from the whole-model relative compute of Table 3. PSNR, SSIM, and LPIPS on these sets are paired against full computation on the same inputs.

C CACHE RELIABILITY WITHIN A SAMPLING RUN

This appendix examines how the reliability of a residual prediction changes within a sampling run and whether the uncertainty score u_i of the trust gate follows this change. On all three backbones,

the per-step median error of the filter's channel-mean prediction at gap one spans more than two orders of magnitude, and the median paired gap-4/gap-1 error ratio exceeds ten (Appendix C.1); on DiT-XL/2, the tensor-level reuse error tends to rise with u_i at a fixed gap (Appendix C.2), and of the three schedules of Figure 1, only the trust gate's computes both steps 48 and 49, where the reuse error jumps (protocol in Appendix C.3). These results demonstrate that the reliability of a residual prediction changes strongly within a run and that, on DiT-XL/2, u_i carries information about this change beyond the gap itself, which motivates gating each step by u_i rather than by a fixed schedule.

C.1 Step and Gap Dependence of the Prediction Error

To quantify how the reliability of a residual prediction depends on the step and the gap, we replay fully computed trajectories (the gate is scored but never acted on) and, for every unprotected step i and gap $\Delta \in \{1, \dots, \Delta_{\max}\}$ for which the filter state Δ steps earlier exists, record the normalized squared error of the channel-mean prediction made from that state, $\sum_c (o_{i,c} - H\widehat{s}_{i|i-\Delta,c})^2 / (\sum_c o_{i,c}^2 + \varepsilon)$, where $\widehat{s}_{i|i-\Delta,c} = A_\Delta s_{i-\Delta|i-\Delta,c}$ is the prediction of Eq. 3 over gap Δ from the filter posterior of step $i-\Delta$ and $H = [1, 0, 0]$ reads its level. Data: for DiT-XL/2, the 100 samples of Appendix C.2 (sample seeds 2000–2099, batch 1) with $\Delta_{\max}$=4; for FLUX.1-dev, the 200 DrawBench prompts with $\Delta_{\max}$=7; for Wan2.1-1.3B, 25 randomly drawn VBench prompts (seed 2, W=15, conditional branch) with $\Delta_{\max}$=5. Table 8 reports the ratio of the largest to the smallest per-step median error at gap 1 and the median gap-4/gap-1 error ratio over matched trajectory–step pairs (number of pairs in parentheses). These statistics describe our own predictor's error, not that of other caching rules. These results show that, on all three backbones, the per-step median error at gap 1 spans more than two orders of magnitude and the median gap-4/gap-1 ratio exceeds ten, which motivates scoring each step by a gap-dependent predicted uncertainty rather than following a fixed schedule.

Table 8: Variation of the filter's normalized squared channel-mean prediction error on fully computed trajectories. Across steps: ratio of the largest to the smallest per-step median error at gap 1. Gap 4 / gap 1: median ratio of the gap-4 to the gap-1 error over matched trajectory–step pairs (number of pairs in parentheses).

Backbone	Across steps (gap 1)	Gap 4 / gap 1
DiT-XL/2	3.3×10^4	18.1 (4,200)
FLUX.1-dev	463	14.4 (8,400)
Wan2.1-1.3B	225	38.4 (775)

C.2 Correlation of the Uncertainty Score with the Reuse Error at a Fixed Gap

On 100 DiT-XL/2 samples (classes drawn with a fixed generator, sample seeds 2000–2099, batch 1 with classifier-free guidance; a set distinct from the DiT-XL/2 control set of Appendix B.3), every step is computed FULL, so the residual $\mathcal{R}_i$ is observed and the filter is updated at every step, while the gate (λ=5) is scored but never acted on. At each unprotected step i and gap $\Delta \le 4$ (the gaps the gate produces at this operating point), we restore the filter posterior of step $i-\Delta$, compute u_i from its prediction over Δ steps, and record the reuse error $\|\mathcal{R}_i - \mathcal{R}_{i-\Delta}\|^2 / \|\mathcal{R}_i\|^2$ over the whole hidden tensor.

At each of the three displayed gaps $\Delta = 1, 2, 3$ (in this shadow evaluation, $u_i \ge \lambda$ at Δ=4 in 4,199 of 4,200 cases), the median reuse error tends to rise with u_i (Figure 4; Spearman $+0.50$, $+0.39$, and $+0.29$ at $\Delta = 1, 2, 3$; the top sextile has $21\times$, $14\times$, and $12\times$ the median error of the bottom sextile). Controlling for Δ, the partial rank correlation between u_i and the reuse error is $+0.34$, whereas that between Δ and the error given u_i is -0.07; at Δ=1 the correlation is positive on all 100 trajectories (mean $+0.40$). These results confirm that, on DiT-XL/2, the tensor-level reuse error tends to rise with u_i even though u_i is computed from channel means, and that u_i carries information about this error beyond the gap itself, which is what the trust gate relies on.

C.3 Protocol of Figure 1: Reuse Error and Latent Drift on DiT-XL/2

Panel (a) of Figure 1 plots, at each unprotected step i, the relative ℓ_2 error $\|\mathcal{R}_i - \mathcal{R}_{i-1}\| / \|\mathcal{R}_i\|$ of reusing the preceding step's residual, averaged over the 100 trajectories of Appendix C.2. With steps

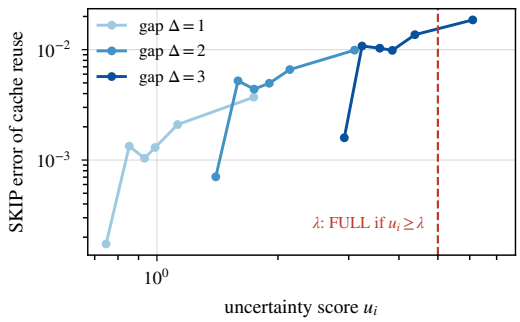


Figure 4: Median reuse error $\|\mathcal{R}_i - \mathcal{R}_{i-\Delta}\|^2/\|\mathcal{R}_i\|^2$ (y-axis) per sextile of the uncertainty score u_i at gaps $\Delta = 1, 2, 3$ on 100 fully computed DiT-XL/2 trajectories (dashed line: trust threshold λ=5).

indexed $1, \ldots, 50$, each of its three schedules has 21 FULL steps: the trust gate's schedule, which is the † schedule of Table 4 (the gate's mask for the first DiT-XL/2 batch, recorded at batch 32; FULL at steps 1–4, 7, 10, 13, 17, 21, 24, 27, 30, 32, 35, 37, 40, 43, 46, 48–50), the uniform interval (1–5, 8, 11, 13, 16, 19, 22, 25, 27, 30, 33, 36, 39, 42, 44, 47, 50), and the input-change-derived schedule of Appendix D.3, shown by its schedule on 14 of 20 pilot batches (1–4, 10, 14, 17, 20, 23, 26, 29, 31, 34, 37, 39, 41, 43, 45, 47, 49, 50); the other six batches differ only between steps 32 and 40, and all 20 skip step 48. For panel (b), each of the same 100 samples (batch 1 with classifier-free guidance) is generated once with full computation and three times on the trust gate's schedule: reusing the last residual, extrapolating it, and with the closed-loop reconstruction, labeled *Reuse*, + *Extrapolation*, and + *Feedback (ours)* in the panel. The drift is $\|x_i - x_i^{\text{full}}\|/\|x_i^{\text{full}}\|$ for the latent x_i after step i, averaged over the samples; after the final step it is 19.1%, 9.1%, and 8.2%. The FID-50K values in both panels are those of Tables 1 and 4, generated with the protocol of Table 1 (50K samples, batch 32, seed 0). Panel (a) shows that, of the three schedules, only the trust gate's computes both steps 48 and 49, where the reuse error jumps, and panel (b) that on this schedule the closed-loop reconstruction ends closer to the full-computation trajectory than both open-loop reconstructions.

D Setting the Trust Threshold and Evaluating the Trust Gate

This appendix explains what the trust threshold λ means and how it is set from a compute target (Appendix D.1), validates this rule prospectively (Appendix D.2), and compares the FULL/SKIP decisions of the trust gate with those of reliability-blind rules at matched budgets (Appendix D.3).

D.1 What the Trust Threshold Is and How It Is Set

Meaning. The trust threshold λ in Eq. 4 is a tolerance on the growth of the filter's predicted uncertainty: an unprotected step is computed once the channel-averaged trace of the covariance predicted from the last FULL checkpoint reaches λ times that of the reference covariance, which is estimated from the warmup residuals of the same run (Appendix A.2). The score u_i thus measures predicted uncertainty relative to how much the residual moved during warmup, and its scale depends on the backbone and the cached tensor. The value of λ is therefore backbone-specific, and only the achieved compute transfers across backbones: λ=5 on DiT-XL/2, λ=1 on FLUX.1-dev, and λ=22 on Wan2.1-1.3B give FLOPs speedups of 2.38×, 2.31×, and 2.19×. Besides the number of SKIP steps, λ sets the blend weight w_i=u_i/λ of Eq. 5, so it also affects how each SKIP step is reconstructed. On Wan2.1-1.3B the gate compares u_i with $\lambda\rho$, so the λ values are nominal (Appendix A.6).

Main-table values. Table 9 lists the λ behind every operating point of Tables 1–3, all obtained by sweeping λ per backbone; the ablations of Table 4 use the 42%-compute point of DiT-XL/2 (λ=5) and the 26%-compute point of FLUX.1-dev (λ=8). The 50%-compute point of Wan2.1-1.3B uses a longer warmup (W=18 instead of 15; Section 4.2), which is why its compute exceeds that of the 46% point despite a larger λ. Figure 5 plots the DiT-XL/2 and FLUX.1-dev operating points against the compared caching methods. At speedups up to that of the fastest *KalmanCache*+ point (2.37× latency speedup on DiT-XL/2, 4.16× FLOPs speedup on FLUX.1-dev), every compared setting has a higher FID-50K (DiT-XL/2) or LPIPS (FLUX.1-dev) than a *KalmanCache*+ point with an equal or

higher speedup. On FLUX.1-dev, the same holds for latency speedup (Table 2) up to 3.44×, that of the fastest *KalmanCache+* point.

Table 9: Operating points of Tables 1–3 and the trust threshold behind each.

Backbone	Operating point (relative compute) → trust threshold λ				
DiT-XL/2	48% → 3	42% → 5	38% → 7	32% → 10	
FLUX.1-dev	43% → 1	36% → 2	28% → 7	26% → 8	24% → 14
Wan2.1-1.3B	50% → 25 (W=18)	46% → 22	42% → 40		

Setting λ on a new deployment. For a fixed configuration (backbone, sampler, resolution, warmup length, batch), λ can be set from a compute target without evaluating output quality: (i) fix a target mean FULL ratio or latency; (ii) starting from λ=1 on a small pilot set disjoint from the evaluation inputs, double λ while the measured compute exceeds the target (halve it if below) until the target is bracketed; (iii) measure two or three values inside the bracket and freeze the one closest to the target before any held-out evaluation. The rule uses only gate and compute logs (plus runtime for a latency target) and was not used to select the main-table points. Because later observations depend on earlier FULL/SKIP decisions, compute is not guaranteed to decrease with λ; the rule relies on it doing so over the tested values, which held in every pilot of the prospective validation in Appendix D.2.

Relation to the control parameters of the compared methods. The compared methods also fix a control parameter per backbone before sampling (an interval N, a number of function evaluations, a threshold l or δ, a parameter K, or the step count); TeaCache, for example, uses l=0.4 and 0.8 on FLUX.1-dev and l=0.1 on Wan2.1-1.3B. Such control parameters differ from model-specific calibration tables, which are curves or coefficients fitted to reference runs of a backbone, such as the rescaling polynomial of TeaCache or the magnitude curve of MagCache. Setting a control parameter offline per configuration is therefore common to the compared methods and the trust gate; what distinguishes the trust gate is what its decision reads during sampling (Appendix D.3).

What the decision reads. The rules differ in what the FULL/SKIP decision reads during sampling: an interval reads only the step index (time-triggered), a change proxy reads a change accumulated since the last FULL step (event-triggered), and the trust gate reads the predicted uncertainty of its own residual prediction (variance-based triggering; Trimpe & D'Andrea, 2014). With fixed noise scales the covariance recursion does not depend on the observed residuals (Section 3.3), so on the image backbones the FULL steps of a run would be fixed at the end of warmup by λ and the warmup statistics (on video, the subsequent values of ρ would also have to be fixed); after warmup, the observed residuals reach the gate only through the online noise-scale updates (Sage–Husa, Appendix A.4; in the ablation without it on the image backbones, the process-noise average of Appendix E.4) and, for video, ρ (Appendix A.6). Appendix D.3 compares the resulting decisions at matched FULL budgets with rules that do not read an uncertainty.

D.2 Prospective Validation of Compute Targeting, Transfer, and Sensitivity

We tested the rule of Appendix D.1 prospectively in two rounds. Round 1 predeclared one compute target per backbone before any pilot run, bracketed λ on pilot inputs disjoint from the evaluation set (20 images for DiT-XL/2, 20 COCO captions for FLUX.1-dev, 10 prompts for Wan2.1-1.3B), refined it inside the bracket, and froze it before the subsequent held-out evaluation, using gate and compute logs only; of the frozen values, only the FLUX.1-dev one (λ=3) had been evaluated on the held-out prompts before, in an earlier threshold sweep. Round 2 added one target each for DiT-XL/2 and FLUX.1-dev, predeclared before any new pilot arm and reusing the round-1 doubling arms (so the frozen DiT-XL/2 value coincided with an existing arm). Every target was met within 2% relative error (Table 10). For the image backbones, the held-out drift lies within or below the interval of the two neighboring main-table thresholds evaluated under the same protocol (DiT-XL/2: the control set of Appendix B.3; FLUX.1-dev: the 200 DrawBench prompts); in the second DiT-XL/2 round it is below both neighbors, so drift is not strictly monotone in λ at this granularity. For Wan2.1-1.3B only the lower neighbor λ=25 was evaluated under the same protocol (the control set at W=15, distinct from the main-table λ=25, W=18 point), and the held-out drift at λ=32 is slightly above it. These results demonstrate that the compute-targeting rule meets every predeclared compute target within 2%, on all three backbones, from gate and compute logs alone, without evaluating output quality.

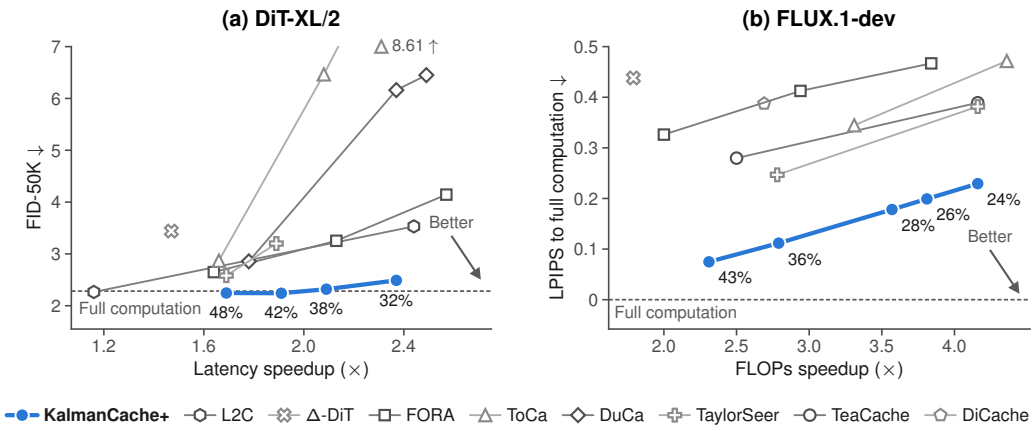


Figure 5: Operating curves on (a) DiT-XL/2 (latency speedup vs. FID-50K) and (b) FLUX.1-dev (FLOPs speedup vs. LPIPS to each method's own full computation; Appendix E.2): *KalmanCache+* (blue, labeled by relative compute) and the compared caching methods (gray, one marker per method; step reduction omitted). Lower right is better.

Table 10: Prospective compute targeting in two rounds. Targets and held-out values are mean FULL ratios (conditional branch for Wan2.1-1.3B); held-out LPIPS is paired against full computation, with the LPIPS of the neighboring thresholds under the same protocol in parentheses; (2nd) marks round 2.

Backbone	Target	Frozen λ	Held-out FULL (rel. err.)	Held-out LPIPS (neighbors)
DiT-XL/2	43.0%	6	42.20% (-1.86%)	0.0383 (λ=5: 0.0255, λ=7: 0.0526)
FLUX.1-dev	33.33%	3	32.81% (-1.57%)	0.1311 (λ=2: 0.1117, λ=7: 0.1782)
Wan2.1-1.3B	44.0%	32	43.92% (-0.18%)	0.0476 (λ=25: 0.0454)
DiT-XL/2 (2nd)	38.0%	8	38.56% ($+1.47\%$)	0.0373 (λ=7: 0.0526, λ=10: 0.0628)
FLUX.1-dev (2nd)	39.0%	1.5	39.36% ($+0.92\%$)	0.0983 (λ=1: 0.0748, λ=2: 0.1117)

A value never evaluated before. The second-round FLUX.1-dev value (λ=1.5) had not been evaluated before it was frozen; on the held-out prompts, which are those of Table 2, it falls between the two main-table points that bracket it (λ=1 and 2, the 43% and 36% points) in both compute (39.36% FULL) and LPIPS (0.0983), so even a threshold that had never been evaluated yields an operating point consistent with its neighbors.

Transfer across prompt sets. On FLUX.1-dev, λ=1 and λ=8 give mean FULL ratios of 43.52% and 26.35% on the 200 DrawBench prompts and 42.53% and 26.29% on the 5,000 COCO captions of Appendix E.2. This stability does not distinguish the gate from the compared rules, whose parameters are also fixed across inputs, but it is what the compute-targeting rule relies on: for both tested thresholds, a frozen λ gives nearly the same compute on a different prompt distribution.

Sensitivity. On the DiT-XL/2 control set, $\lambda \in \{3, 5, 7, 10\}$ gives mean FULL ratios of 55.3%, 47.0%, 40.0%, and 34.3% with paired LPIPS of 0.0175, 0.0255, 0.0526, and 0.0628: over these values, larger λ reduces compute and increases drift. These per-image-gating ratios exceed the batch-32 relative compute of the same λ in Table 9 (cf. Appendix E.1). On FLUX.1-dev, over 19 values of λ from 1 to 200 on the 200 DrawBench prompts, the Spearman correlation is 0.998 between λ and LPIPS and -0.989 between λ and latency. Over these values, λ therefore acts as a single knob that trades fidelity for compute on both image backbones.

Pilot size and cost. In a post-hoc resampling of existing runs (20,000 random 20-input subsets, the same inputs for every λ), 95% of subsets estimate the whole-set mean of every candidate within a relative error of 3.49% for the FULL ratio (latency: 2.57%) on the DiT-XL/2 control set ($\lambda \in \{3, 5, 7, 10\}$) and of 4.92% for latency on the 200 DrawBench prompts (the five FLUX.1-dev values of Table 9). Running these candidates on such pilots takes about 0.8 and 16 A100-minutes. These post-hoc resampling results support 20-input pilots, which take minutes of GPU time, for estimating the FULL-ratio or latency means of the tested candidates.

D.3 Comparison of FULL/SKIP Decision Rules at Matched Budgets

This subsection extends the decision comparison of Table 4. We test whether, at the same FULL budget, the steps chosen by the trust gate give better outputs than those chosen by rules that do not read an estimate of how uncertain the cached prediction is and can therefore skip the steps where the cache is least reliable. In each comparison the interval and the proxy threshold are set to reach the budget of the gate's main-table λ, which fixes the offline setting of every rule, and, except in the plain-reuse check below, every decision variant uses the closed-loop reconstruction with the gate's λ (on a SKIP step with $u_i \geq \lambda$ its weight is clipped at one), so the variants differ only in which steps they compute. The comparisons are made on the two image backbones at the budgets of Table 4 (21 of 50 FULL steps on DiT-XL/2, 13 of 50 on FLUX.1-dev), and the conclusions below are stated for these budgets and the tested rules. The input-change-derived schedule refreshes once the accumulated relative change of the first block's modulated input (the signal of TeaCache) reaches a threshold set only by the FULL count on a 20-batch or 20-prompt pilot, with no model-specific rescaling polynomial. On the 20-batch DiT-XL/2 pilot, the adaptive gate averages 21.1 FULL steps and the input-change-derived schedule 21.2; on FLUX.1-dev, over the 200 DrawBench prompts, both average 13.2.

The gate's schedule and the two rules. Replaying the schedule that the gate chose for the first DiT-XL/2 batch or FLUX.1-dev prompt on every input († in Table 4) fixes the decision before sampling (21 and 13 FULL steps), and at the approximately matched budgets of Table 4 it still outperforms both rules: FID-50K 2.251 against 2.454 (uniform interval) and 2.295 (input-change-derived schedule) on DiT-XL/2, and on FLUX.1-dev +1.57 dB PSNR [1.42, 1.73] and +4.36 dB [4.06, 4.68] (paired 95% CIs; the SSIM and LPIPS CIs also exclude zero). On FLUX.1-dev, the adaptive gate exceeds the uniform interval by +2.26 dB [2.03, 2.50], +0.059 SSIM, and -0.074 LPIPS, and the input-change-derived schedule by +5.05 dB [4.70, 5.42], +0.201, and -0.256. The † schedule's advantage over the uniform interval on FLUX.1-dev is not attributable to different locations of the final two FULL steps (steps 44 and 50), which these two schedules share. With plain reuse, whose reconstruction reads no uncertainty score, the † schedule still exceeds the uniform interval on FLUX.1-dev at seed 0 by +0.84 dB PSNR [0.75, 0.93], +0.024 SSIM, and -0.032 LPIPS (13 FULL steps each; the SSIM and LPIPS CIs also exclude zero). The replay comparison supports the gate's placement at these budgets; it does not isolate a benefit of the online uncertainty updates. Both rules decide from the step index or the input change alone, without reading how reliable the cache is; on the DiT-XL/2 pilot batches, both skip step 48, where the reuse error jumps, whereas the gate's schedule computes it (Appendix C.3). These results confirm that, at these approximately matched budgets, placing FULL steps by predicted uncertainty outperforms both tested reliability-blind rules on both image backbones, even when the gate's schedule is fixed before sampling.

Second seed. On DiT-XL/2, where FID-50K has no paired CI, a second sampling seed preserves the ordering of the four decision variants (FID-50K: uniform interval 2.395, input-change-derived schedule 2.251, gate's schedule 2.195, adaptive gate 2.187). At FLUX.1-dev seed 1 (paired against full computation at that seed), with λ=8, the proxy threshold, and the † schedule frozen at their seed-0 values, the adaptive gate (13.13 FULL steps) exceeds the uniform interval (13) by +2.33 dB PSNR [2.07, 2.61], +0.054 SSIM, and -0.073 LPIPS, the input-change-derived schedule (13.34) by +5.06 dB [4.67, 5.48], +0.206, and -0.274, and the † schedule (13) by +0.75 dB [0.54, 0.97], +0.011, and -0.014; all SSIM and LPIPS CIs also exclude zero. At this seed the four decision variants reach PSNR/SSIM/LPIPS of 19.37/0.775/0.256 (uniform interval), 16.64/0.623/0.457 (input-change-derived schedule), 20.96/0.818/0.197 († schedule), and 21.70/0.829/0.183 (adaptive gate), the same ordering in all three metrics as at seed 0 in Table 4. On both image backbones, the ordering of the four decision variants, including the adaptive gate's advantage over both rules and over its own fixed schedule, is therefore not specific to seed 0.

TeaCache's rescaling. The input-change-derived schedule accumulates the raw relative change, without the model-specific rescaling polynomial of TeaCache. With that polynomial for FLUX.1-dev applied to the same signal and TeaCache's accumulate-and-reset rule (same protected steps and closed-loop reconstruction as the other decision variants; threshold 0.925, set from compute logs only to reach the budget), the rule computes 13 FULL steps on every prompt (8 distinct schedules over the 200 prompts), and at seed 0 the adaptive gate exceeds it by +3.14 dB PSNR [2.87, 3.43], +0.087 SSIM, and -0.118 LPIPS; the SSIM and LPIPS CIs also exclude zero. Without any calibration table,

the adaptive gate thus also outperforms TeaCache's rescaled decision rule on FLUX.1-dev at this budget and seed.

Per-prompt schedules at a fixed λ. At λ=8 on FLUX.1-dev the gate produces 21 distinct schedules for the 200 prompts (13 FULL steps on 165 prompts and 14 on 35; the most frequent schedule occurs on 51). The 0.69 dB between the adaptive gate and the † replay in Table 4 is the mean per-prompt gain over the 200 prompts: +0.67 dB on the 154 prompts with 13 FULL steps and a schedule other than †, +0.97 dB on the 35 prompts with 14 (one more FULL step than †), and zero on the 11 prompts whose own schedule is †, whose outputs are bit-identical to the † run. Weighted by prompt count, the first two groups contribute 0.52 dB and 0.17 dB of the 0.69 dB. Most of this gain thus comes from the 154 prompts on which the gate computes as many FULL steps as the † schedule but places them differently, and less from the 35 prompts with one more FULL step.

Gate without Sage–Husa adaptation. The matched-budget comparison of the gate with and without the Sage–Husa update on FLUX.1-dev (Appendix E.4; λ=2 versus λ=8 at about 18 FULL steps) is between two gates that both read a predicted uncertainty but differ in how its noise scales are re-estimated from the innovations and, through $w_i{=}u_i/\lambda$, in the blend weights, so this comparison does not isolate the FULL/SKIP decision and is reported with the feedback components in Appendix E.4. The threshold that reaches the budget differs (2 versus 8) because the scores differ, which is also why values of λ are not comparable across configurations.

E Robustness, Feedback Ablations, and Overhead

This appendix checks how the results depend on the batch size, the seed, the prompt distribution, the fixed filter constants, and the warmup length, and whether raising the guidance scale under full computation reaches the FID-50K of the 42%-compute DiT-XL/2 point (Appendices E.1–E.3), ablates the blend weight and the noise adaptation (Appendix E.4), and measures the runtime and memory overhead (Appendix E.5).

E.1 Effect of Batch Size on the Shared Gate

On DiT-XL/2 the uncertainty score is computed over the CFG-expanded batch, so the images of a batch share one FULL/SKIP schedule. On a fixed pool of 128 ImageNet images at λ=5 (the 42%-compute point), each cached arm is paired with a full-computation reference generated in the same batch. With sample batch sizes 1, 8, and 32, the mean FULL ratio is 46.9% (per-image range 42–66%), 42.9% (42–46%), and 42.0% (identical for every image), and the paired PSNR/SSIM/LPIPS are 34.10/0.9568/0.0298, 32.75/0.9485/0.0360, and 32.63/0.9486/0.0358. Regrouping the pool into different batches of 32 leaves the cached outputs close to the original ones (PSNR 48.29 dB, SSIM 0.9941, LPIPS 0.0032), whereas an exact repeat reproduces all 128 outputs, so this difference comes from batch composition. (Absolute values reflect this pool and its same-batch references and differ from those of the DiT-XL/2 control set of Appendix B.3: LPIPS 0.0298 versus 0.0255 at λ=5.) Batch size thus shifts the operating point of the shared gate, whereas batch composition changes its outputs only slightly; the main-table results use batch 32, and a deployment should select λ at its intended batch size.

E.2 Distributional, Multi-Seed, and Guidance Checks

Distributional check. With 5,000 FLUX.1-dev images per arm generated from COCO2017-val captions (the same caption and seed for each image in every arm) and FID/KID computed against the same 5,000 real images (clean-fid, Inception-v3), full computation gives FID 35.38 / KID 0.01311, the 43% point (λ=1) 34.85 / 0.01273, and the 26% point (λ=8) 34.68 / 0.01238. With a spread of 0.70 FID and 0.0007 KID across the three arms, from which we claim neither improvement nor equivalence, the observed FID and KID point estimates on COCO captions are lower at both cached points than for full computation. We report no Fréchet video distance (FVD) for video because the VBench prompt protocol has no canonical real-video reference; rather than building a nonstandard reference distribution from generated full-computation videos, we use the VBench score, which aggregates 16 reference-free dimensions, and paired PSNR/SSIM/LPIPS.

Four-seed extension. For FLUX.1-dev, paired metrics (PSNR, SSIM, and LPIPS) in Table 2 are computed against a full-computation run of the implementation in which each method was run; for Wan2.1-1.3B, all methods share one full-computation reference. The TeaCache comparisons below score both arms against the same reference for each prompt and seed. Over the 200 DrawBench prompts and seeds $\{1, 2, 3, 4\}$ (800 paired outputs per arm; 95% intervals from a crossed seed–prompt bootstrap), the 26% point (λ=8) has paired PSNR 21.26 dB [20.72, 21.83], SSIM 0.816 [0.800, 0.832], and LPIPS 0.193 [0.178, 0.207], and the 24% point (λ=14) 20.43 dB [19.89, 21.02], 0.793 [0.775, 0.811], and 0.223 [0.207, 0.239]; the 95% intervals of the CLIP Score and ImageReward differences to full computation include zero at both points (CLIP Score $+0.007$ [-0.180, $+0.189$] and -0.010 [-0.255, $+0.209$]; ImageReward -0.002 [-0.027, $+0.022$] and -0.007 [-0.042, $+0.024$]). Against TeaCache (l=0.8; 4.16× FLOPs at the Table 2 seed, 3.93× averaged over these seeds), regenerated for the same seeds and scored against the same references, the 24% point improves PSNR by $+4.64$ dB [4.19, 5.09], SSIM by $+0.116$ [0.104, 0.128], and LPIPS by -0.180 [-0.195, -0.165], with the same sign on every seed (per-seed PSNR gains 4.18–4.89 dB); the CLIP Score ($+0.12$ [-0.25, $+0.48$]) and ImageReward ($+0.061$ [-0.0003, $+0.120$]) differences are not significant. At the Table 2 seed (both arms scored against the same full-computation run), the same comparison over the 200 prompts (95% intervals from a paired-prompt bootstrap) gives ImageReward $+0.143$ [0.067, 0.225], CLIP Score $+0.577$ [0.108, 1.103], PSNR $+4.41$ dB [3.98, 4.85], SSIM $+0.105$ [0.095, 0.115], and LPIPS -0.160 [-0.173, -0.146]. These results confirm that the paired-fidelity advantage of the 24%-compute point over TeaCache (l=0.8) is not an artifact of a single seed or of per-method references.

Guidance control (DiT-XL/2). We test whether raising the CFG scale under full computation reaches the FID-50K of the 42%-compute point in Table 1. Full computation at CFG scales 1.6 and 1.7 gives FID-50K 2.629 and 3.235 against 2.282 at the CFG scale 1.5 used throughout; raising the guidance scale therefore does not reproduce the FID of 2.241 reached at the 42%-compute point.

E.3 Sensitivity to the Fixed Filter Constants and the Warmup Length

Fixed filter constants. α_{floor}=0.05 keeps the Sage–Husa rate α_i from decaying toward zero as N_i grows (Appendix A.4), and q_b=0.01 lets the bias b drift with one hundredth of the trend's process noise (Appendix A.1); both are fixed defaults that were never retuned. On the DiT-XL/2 control set of Appendix B.3 (λ=5), $\alpha_{\text{floor}} \in \{0.01, 0.05, 0.15\}$ gives mean FULL ratios of 47.0%, 47.0%, and 46.7% with PSNR/SSIM/LPIPS of 34.42/0.9595/0.0255, except 34.44/0.9596/0.0255 at 0.15; $q_b \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ gives FULL ratios of 46.8%, 46.9%, 47.0%, 47.8%, and 48.7% with 34.41/0.9591/0.0259, 34.42/0.9590/0.0261, 34.42/0.9595/0.0255, 35.53/0.9661/0.0210, and 37.18/0.9732/0.0163, so values below the default leave the operating point nearly unchanged, whereas q_b=0.05 and 0.1 move it, trading 0.8 and 1.7 points more compute for higher fidelity. At $\lambda \in \{3, 7, 10\}$ on the same set, $\alpha_{\text{floor}} \in \{0.01, 0.05\}$ gives bit-identical outputs, and 0.15 changes PSNR, SSIM, LPIPS, and the FULL ratio by at most 0.04 dB, 0.00021, 0.00005, and 1.08 points. On FLUX.1-dev (first 100 DrawBench prompts, seed 1, λ=8), $\alpha_{\text{floor}} \in \{0.01, 0.05, 0.10, 0.15\}$ gives bit-identical outputs at 26.18% FULL (the floor only acts once $2/(N_i + 1)$ falls below it), and $q_b \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ changes PSNR, SSIM, LPIPS, and the FULL ratio relative to the defaults by at most 0.14 dB, 0.002, 0.003, and 0.4 points. Except at $q_b \in \{0.05, 0.1\}$ on DiT-XL/2, which move the operating point, both filter constants thus change the outputs of the two image backbones only marginally, which supports retaining the defaults in the tested configurations.

Warmup length. On the Wan2.1-1.3B control set (λ=25), reducing W below 15 reduces compute and worsens all three paired fidelity metrics (Table 11); W=15, the main-table default, has the best SSIM and LPIPS point estimates, and the SSIM and LPIPS differences of W=19 and W=23 from it have 95% intervals that include zero. PSNR, in contrast, keeps rising with W: its differences from W=15 are $+1.22$ dB [-0.13, $+2.26$] at W=19 and $+2.05$ dB [0.54, 3.34] at W=23. Shorter warmups thus save compute at a loss in all three fidelity metrics, whereas longer ones add compute for a PSNR gain, significant only at W=23, but no significant SSIM or LPIPS gain, which makes W=15 a balanced default on Wan2.1-1.3B.

Tuning guide. (i) Keep α_{floor} and q_b at their defaults. (ii) Set W: the image backbones use W=4, the smallest warmup for which the initialization of Appendix A.2 is defined (the unbiased variance of the second differences needs four observations); for a new video backbone, choose the smallest

W that meets a predeclared drift limit against full computation. (iii) Set λ at the intended batch size (Appendix E.1) from a compute target with the rule of Appendix D.1, validated in Appendix D.2.

Table 11: Warmup sensitivity on the Wan2.1-1.3B control set (50 prompts, seed 2, λ=25; W=15 is the main-table default; FULL ratio of the conditional branch; PSNR/SSIM/LPIPS paired against full computation).

W	7	11	15	19	23
Latency (s)	68.33	79.92	90.94	103.59	116.32
FULL ratio	31.4%	38.0%	44.1%	51.0%	57.8%
PSNR ↑	21.05	26.79	29.26	30.47	31.31
SSIM ↑	0.756	0.886	0.920	0.911	0.914
LPIPS ↓	0.196	0.070	0.045	0.061	0.063

E.4 Effect of the Blend Weight and Noise Adaptation

Blend weight. On FLUX.1-dev, replacing the blend weight $w_i = \min(1, u_i/\lambda)$ with full weight (w_i=1 in Eq. 5) on the † schedule of Table 4 lowers PSNR from 20.85 to 18.42 dB and SSIM from 0.794 to 0.684 and raises LPIPS from 0.216 to 0.376 (paired 95% CIs exclude zero). Full weight applies the extrapolation and the C-DOB correction in full at every SKIP step, whereas the blend weight scales them by the predicted uncertainty, capped at one. On FLUX.1-dev, weighting the correction by u_i therefore improves all three fidelity metrics over applying it in full.

Sage–Husa noise adaptation. Without the Sage–Husa update, the DiT-XL/2 and FLUX.1-dev filters keep the measurement noise at its warmup value while the process noise follows an exponential moving average of the squared innovations, and the same threshold lands at more FULL steps: 25.1 instead of 21.1 on a 20-batch DiT-XL/2 pilot and 18.1 instead of 13.2 on FLUX.1-dev. At a matched budget of about 18 FULL steps on FLUX.1-dev (the 36%-compute point of Table 2; λ=2 with the update at 17.9 steps, λ=8 without at 18.1), the update raises PSNR from 21.96 to 25.41 dB (paired 95% CI of the gain [2.99, 3.93]) and SSIM from 0.827 to 0.883 and lowers LPIPS from 0.177 to 0.112, so on FLUX.1-dev, re-estimating the noise scales from the innovations improves fidelity at a matched budget. In the Wan2.1-1.3B variant without the update, both noise scales are instead frozen at their per-prompt post-warmup values, so results are compared only within each backbone. On the Wan2.1-1.3B control set (λ=25, W=15), freezing raises the conditional-branch FULL ratio from 44.1% to 45.2% and worsens PSNR (29.26 to 28.56 dB, paired 95% CI [-1.51, -0.01]) and LPIPS (0.0454 to 0.0568, [$+0.0009$, $+0.0268$]); the SSIM change (0.9203 to 0.9062, [-0.035, $+0.0004$]) is not significant. On Wan2.1-1.3B, freezing thus adds compute and worsens PSNR and LPIPS.

E.5 Runtime and Memory Overhead of KalmanCache+

We recorded the FULL/SKIP schedule chosen by the gate at the 42%-compute DiT-XL/2 point (λ=5, batch 32) and forced it in two arms, plain residual reuse and full *KalmanCache+* (gating, state and noise updates, extrapolation, blending, C-DOB). Over 20 paired, order-balanced blocks in one exclusive-GPU session, *KalmanCache+* costs 2.12% more sampling-loop wall time, excluding VAE decoding (95% paired-block bootstrap CI [2.09, 2.15]; CUDA-event timing agrees); this is the cost of the cache path at an identical mask, not an end-to-end speedup. Its peak allocation, measured in the same session, exceeds that of plain reuse by 216.1 MiB (below 0.3% of the 80-GiB device). Its total persistent state after warmup, counted from the tensor shapes, is the residual memory, i.e., the last two FULL residuals and the stored mismatch (72.0 MiB each), plus 0.10 MiB of estimator state, excluding transient scratch. On FLUX.1-dev the same state takes 96.3 MiB (about 0.12% of the device; the two residuals are kept in 16-bit precision). The per-channel state algebra is $O(C)$, whereas transformer FLOPs grow roughly quadratically with width; the tensor-level path is linear in the residual size. On Wan2.1-1.3B, an all-FULL run of the wrapped model (FULL-side bookkeeping only) is 0.53% slower than the unmodified model over 50 paired prompts (95% CI [0.31, 0.77]). These results show that the closed-loop machinery of *KalmanCache+* is lightweight: at an identical DiT-XL/2 mask it adds 2.12% sampling-loop time and less than 0.3% of device memory over plain reuse, and this cost is included in all reported latencies.

F LIMITATIONS

The three-dimensional linear state models gradual residual dynamics; an abrupt change of the residual is absorbed only at the next FULL step rather than anticipated. The warmup steps are computed in full and are not amortized, so the achievable speedup falls as the sampler gets shorter. On DiT-XL/2 the images of a batch share one gate, so the operating point should be selected at the intended batch size (Appendix E.1). Like the intervals and thresholds of the compared methods, the trust threshold must be set once per deployment configuration, and its value does not transfer across backbones; the rule of Appendix D.1 sets it without evaluating output quality (Appendix D.2), but it targets compute, not a drift level. Finally, each main-table result uses one sampling seed; multi-seed evidence covers two FLUX.1-dev operating points (Appendix E.2) and the four decision variants of Table 4 on both image backbones, including the DiT-XL/2 42%- and FLUX.1-dev 26%-compute points (Appendix D.3). Automating the compute budget and per-sample gating for heterogeneous batches is left to future work.