WHEN TO RECOMPUTE: ERROR-REGULATED ADAPTIVE CACHING FOR DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion transformers have become the dominant backbone for high-fidelity image and video generation, yet their inference remains prohibitively expensive: each sample requires dozens of full network evaluations executed under strict temporal dependencies that forbid parallelization. Feature caching alleviates this cost by reusing or predicting intermediate features across denoising steps, but existing methods apply a fixed, sample-agnostic caching schedule and provide no means of detecting when a cheap prediction has drifted, so prediction errors compound and quality collapses as the acceleration ratio grows. We present *Error-Regulated Adaptive Caching* (ERAC), an acceleration framework that regulates cache reuse by an explicit error budget, which adapts on the current sample and timestep and bounds the drift during reuse. Our method organizes the denoising trajectory into computation segments that open with a full evaluation and continue with cached predictions, while a lightweight, time-aware autoregressive estimator monitors the cumulative drift of the cached trajectory and triggers a rollback to a full computation whenever the estimated error exceeds a prescribed threshold. Because the trigger adapts per sample and per timestep, computation is reallocated to the steps and samples that actually need it, keeping expensive forward passes sparse while bounding trajectory error within a controllable tolerance. On FLUX.1-dev, our method attains an Image Reward above 0.9736 at 6.04 times acceleration, which reduces the deviation from the full evaluation by more than a factor of three against the strongest fixed-schedule baseline, and it still holds at 7.08 times acceleration where such baselines break down. On Wan2.1 1.3B it reaches the best PSNR, LPIPS, and SSIM among all compared methods at speedups up to 3.04 times.

1 INTRODUCTION

Diffusion models have become a dominant paradigm for high-fidelity image and video generation, and the adoption of Diffusion Transformers (DiTs) has further advanced visual generation quality through scalable transformer architectures. However, these models remain constrained by an intrinsic limitation of diffusion sampling. The strict sequential dependency between denoising steps requires dozens to hundreds of them to be executed one after another without parallelization Liu et al. (2025b). This cost is compounded by the fact that each step demands a complete forward pass through an increasingly large network, so inference overhead escalates sharply as models grow in scale and generation tasks become more sophisticated Zheng et al. (2025). For instance, modern text-to-image models such as FLUX.1-dev run a 12-billion-parameter network across many timesteps for a single sample, making real-time and large-scale deployment computationally prohibitive.

To alleviate this cost, numbers of acceleration methods have emerged, among which training-free approaches are particularly attractive because they incur no additional training compute and preserve the diversity of the original model's outputs. Existing training-free methods roughly fall into two families. The step reduction method uniformly removes denoising steps, such as uniform step skipping and Δ-DiT Chen et al. (2024). The feature caching method reuses or extrapolates intermediate features across steps, e.g., FORA Selvaraju et al. (2024), TeaCache Liu et al. (2025a), and TaylorSeer Liu et al. (2025b). Both families, however, share a key weakness. Step reduction applies the same pruning rule everywhere and therefore cannot distinguish the importance of different denoising stages, which often causes severe quality degradation. On the other hand, feature caching

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.

is unstable as well, because its cache schedule is a fixed interval that must be hand-tuned. Once this interval grows, the reused features drift away from the true trajectory while the accumulated error grows entirely unnoticed. The core issue is that neither family quantifies the magnitude of the current error. As a result, the decision about when to stop reusing features and instead perform a full computation is made by a hard-coded table rather than by the actual state of the sample at a given timestep.

We argue that the quality loss of cache-based acceleration is, in essence, an uncontrolled accumulation of error, and one should therefore estimate and monitor this error directly instead of relying on a fixed interval. This motivates our core idea. At every cached step, we predict the current accumulated drift, and whenever it exceeds a budget threshold, we roll back to a full computation that rebuilds a reliable reference. This yields a *predict-and-rollback* closed loop in which expensive full computations are spent only where the error genuinely becomes large, which gives adaptive, on-demand acceleration.

Concretely, ERAC partitions the denoising trajectory into segments, each opening with a full computation and continuing with several cheap cached predictions. A time-aware estimator reads a compact four-dimensional feature of every cached step. The relative drift and the drifted time describe the step itself, while the segment phase records how many cached steps have already passed inside the current segment. The accumulated error feeds the output of the previous step back into the current prediction, and each score depends on the errors reported earlier in the same segment rather than on the current step alone. This autoregressive design explicitly conditions the prediction on how far the current segment has already deviated, and any score above the threshold triggers a rollback. On a rollback, both the reference state and the accumulated error are reset and a new segment begins. This loop relies on an estimator that is far cheaper than one network forward pass, and it remains training-free. It is fitted offline on a small set of calibration trajectories and then frozen, thus no gradient is ever propagated through the diffusion backbone.

Our contributions are summarized as follows:

- We propose **ERAC**, a training-free acceleration framework that models cache reuse as a data-driven *predict-and-rollback* error-feedback loop, replacing hand-crafted fixed cache schedules with adaptive rollback decisions.
- We design a **time-aware autoregressive error estimator** that accurately predicts the accumulated drift at each cached step, thereby triggering full computation precisely when the error exceeds the budget. The estimator is trained offline and frozen at inference, and its cost at each cached step stays below 0.1% of a full forward pass.
- Extensive experiments on **FLUX.1-dev** text-to-image generation and **Wan2.1 1.3B** text-to-video generation show that ERAC incurs substantially less quality loss than state-of-the-art baselines at higher speedups. On FLUX.1-dev, ERAC keeps an Image Reward of 0.9736 at $6.04\times$ acceleration, cutting the deviation from the full evaluation by more than a factor of three, and on Wan2.1 1.3B it reaches the best PSNR, LPIPS, and SSIM at up to $3.04\times$.

2 Relative works

2.1 Diffusion Models and Inference Cost

Diffusion models have emerged as a powerful paradigm for high-quality image and video generation through iterative denoising Ho et al. (2020); Rombach et al. (2022); Peebles & Xie (2023); Esser et al. (2024). Meanwhile, they have rapidly scaled to billion-parameter architectures, substantially increasing the computational cost of each denoising evaluation. For example, DiT-XL/2 contains approximately 2.7B parameters, Stable Diffusion 3 scales to 8B parameters, and FLUX.1-dev contains 12B parameters Peebles & Xie (2023); Esser et al. (2024); Labs (2024). The computational burden becomes even more pronounced for video generation. Wan2.1 provides 1.3B and 14B variants, while HunyuanVideo employs a foundation model with over 13B parameters Wan et al. (2025); Kong et al. (2024). Since these models must be evaluated repeatedly along the denoising trajectory, even a single generation can require substantial computation. Even for the lightweight Wan2.1 1.3B model, the large number of video latents makes a single inference demand about 164 TFLOPs of

computation, which introduces severe latency. Consequently, reducing redundant model evaluations while preserving generation quality has become a critical direction for efficient diffusion inference.

2.2 Training-Free Diffusion Acceleration

Training-free acceleration methods aim to reduce diffusion inference cost without modifying the original model parameters. Existing approaches can be broadly divided into sampling-step reduction and feature reuse or forecasting.

Sampling-step reduction. A straightforward strategy is to reduce the number of denoising evaluations by skipping selected sampling steps or designing more efficient sampling schedules. For example, Δ-DiT investigates the redundancy of different DiT blocks across the sampling trajectory and introduces a training-free caching mechanism that dynamically allocates computation to different blocks at different stages Chen et al. (2024). EPSS further explores empirically pruning redundant sampling steps in flow-matching-based generation Zheng et al. (2025). These approaches demonstrate that substantial computation can be removed from the denoising trajectory without retraining the underlying model. However, their scheduling strategies are generally determined by predefined sampling rules or empirically derived criteria, which do not explicitly account for the approximation error induced by each individual skipping decision. Consequently, a schedule that is effective on average may become suboptimal for individual samples or different regions of the generation trajectory.

Feature caching and temporal forecasting. Another line of work exploits the temporal redundancy of intermediate features across adjacent denoising steps. FORA Selvaraju et al. (2024) and DeepCache Ma et al. (2024) reuse intermediate representations computed at previous timesteps to avoid redundant computation. Beyond simple reuse, TaylorSeer Liu et al. (2025b) introduces a forecasting-based paradigm that predicts future features from previously computed features using Taylor expansion. While these methods can provide substantial acceleration, their cache schedules are largely determined by predefined reuse intervals or manually selected configurations. When the temporal interval becomes large, the cached representation may deviate increasingly from the representation that would have been obtained through full computation, while such deviation is not explicitly estimated during inference.

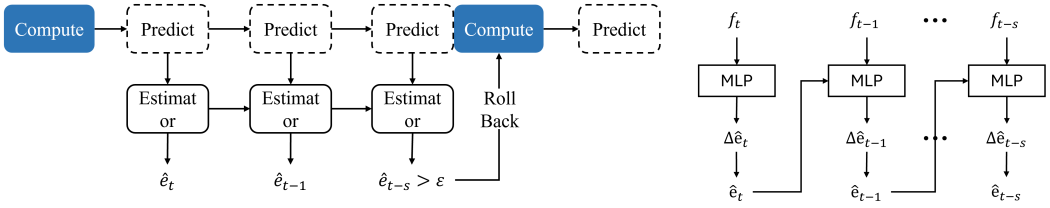
More recent methods introduce dynamic criteria to determine whether cached features should be reused. TeaCache Liu et al. (2025a) estimates the variation of model outputs using timestep-embedding-aware input differences and dynamically determines cache decisions. ToCa Zou et al. (2025) further introduces token-wise feature caching by considering the heterogeneous temporal redundancy of different tokens. SeaCache Chung et al. (2026) exploits spectral evolution to construct a content-aware dynamic cache schedule. SpeCa Liu et al. (2025c) adopts a speculative forecasting-and-verification strategy to reject unreliable predictions. These methods demonstrate that cache decisions should depend on the characteristics of the current denoising trajectory rather than relying solely on a globally fixed interval. Nevertheless, their dynamic criteria are based on specific proxy signals, such as input differences, token-wise redundancy, or spectral evolution. These proxy signals are correlates of the true error rather than the error itself. Such signals do not always reflect the actual error faithfully, which can make the resulting cache schedule less dependable when the trajectory drifts far from the full-computation path.

3 Methodology

3.1 Preliminary

Diffusion Models. Diffusion models generate a sample by reversing a gradual corruption process. A pure noise vector becomes a structured signal through a long chain of denoising steps. Let a_t and b_t be the signal retention factor and the injected variance at denoising time t, and let $\bar{a}_t$ be the cumulative product of a up to t. A reverse step samples the next latent from a Gaussian,

$$q_\phi(z_{t-1} \mid z_t) = \mathcal{N}\left(z_{t-1}; \frac{1}{\sqrt{a_t}}\left(z_t - \frac{1-a_t}{\sqrt{1-\bar{a}_t}}\,\hat{\nu}_\phi(z_t, t)\right), b_t I\right), \tag{1}$$



(a) Predict-and-Rollback mechanism (b) Error Estimator

Figure 1: Illustration of the proposed ERAC framework. (a) The adaptive predict-and-rollback mechanism. The denoising trajectory is split into segments, each opening with a full computation and continuing with cached predictions, and a rollback is fired when the estimated error passes the threshold, which resets the reference and starts a new segment. (b) The time-aware autoregressive error estimator. Features are read from the cached trajectory, including the error reported at the previous cached step, so that each score is conditioned on the errors accumulated earlier in the same segment, and a shared small network maps them to the score that drives the rollback.

where z_t is the noisy latent at time t and $\hat{\nu}_\phi(z_t, t)$ is the noise prediction of the model with parameters ϕ. Each step requires one full evaluation of the network. The steps follow a strict order, thus the total cost scales linearly with the number of evaluations and cannot be reduced by parallel execution. This cost is the main obstacle that acceleration methods must remove.

Diffusion Transformer. A Diffusion Transformer (DiT) Peebles & Xie (2023) replaces the convolutional backbone with a stack of transformer blocks. The whole network is a composition of B blocks,

$$\Phi \;=\; \phi_B \circ \phi_{B-1} \circ \cdots \circ \phi_1, \tag{2}$$

where each block ϕ_j combines a self attention sublayer, a cross attention sublayer that injects the conditioning signal, and a feed forward sublayer. The latent is first split into P patches and projected into a token sequence $z^t = \{z_p^t\}_{p=1}^P$, and the time index t is injected through adaptive layer normalization, so that every block modulates its own activations according to the current noise level. The attention and feed forward operations dominate the cost of a forward pass, which means that reusing an activation at a neighboring time can remove a large share of the computation while keeping the output close to the full evaluation.

TaylorSeer as the Prediction Model. Our method treats the noise or velocity prediction as a black box and only intervenes on the intermediate activations, so it can be paired with any cache predictor. In all experiments we adopt TaylorSeer Liu et al. (2025b) as the prediction model, because it is training-free and it can extrapolate several steps ahead from a single full evaluation. Let h_j^t denote the activation of block j at time t, let T be the stride between two consecutive full evaluations, and suppose that the most recent full evaluation was performed at time t. TaylorSeer approximates the activation at a later time $t - dT$ by a finite difference expansion around t,

$$\tilde{h}_j^{t-dT} = h_j^t + \sum_{i=1}^{O} \frac{\delta^i h_j^t}{i!\, T^i} (-d)^i, \tag{3}$$

where O is the expansion order, d is the number of strides between the current time and the reference time, and $\delta^i h_j^t$ is the i th order forward difference of the activation,

$$\delta^i h_j^t = \sum_{r=0}^{i} (-1)^{i-r} \binom{i}{r} h_j^{t+rT}. \tag{4}$$

The difference terms reuse activations from previous full evaluations, so the expansion adds no training and no extra network pass. The error grows with the stride d and stays nonzero because the order O is finite, and it peaks when the activation curve bends sharply. TaylorSeer therefore fixes T and O in advance and applies the same schedule to every sample. Our method keeps the predictor but drops this rigidity, since the error estimator in the next subsection decides when to reuse a predicted activation instead of a preset stride.

3.2 Adaptive Predict-and-Rollback Mechanism

As illustrated in Figure 1(a), we formulate the accelerated denoising trajectory as a sequence of computation *segments*, where each segment opens with a full network evaluation and continues with a number of cheap cached predictions. Given a denoising schedule $\{t_0 > t_1 > \cdots > t_N\}$, we partition it into non-overlapping segments G_k, and the segment boundary is exactly where a full computation re-establishes a reliable reference state,

$$G_k = \{\, t_{s_k}, \ldots, t_{s_{k+1}-1} \,\}, \qquad x_{\mathrm{ref}}^{(k)} = x_{t_{s_k}}, \tag{5}$$

where $x_{t_{s_k}}$ is the latent produced at the opening full computation step of segment k. To keep the quality of the cached trajectory under control, a lightweight estimator attached to the pipeline produces, for every cached step t, an instantaneous error score $\hat{e}_t$ that reflects how far the cheap prediction has drifted from the segment reference. The score is obtained from a parametric model g_θ with learnable parameters θ,

$$\hat{e}_t \;=\; g_\theta(\mathbf{f}_t)\,, \qquad r_t \;=\; \mathbf{1}\big[\,\hat{e}_t > \varepsilon\,\big], \tag{6}$$

where $\mathbf{f}_t$ is the feature vector computed at step t from the segment reference $x_{\mathrm{ref}}^{(k)}$, and $r_t \in \{0, 1\}$ is the rollback indicator. Before each cached step is executed, we trigger a rollback once the estimated accumulation exceeds a user-specified threshold. Within a segment, every timestep plays one of the following roles:

- **Opening step.** The first step of the segment, $t = t_{s_k}$, is always a full computation.
- **Cached step.** Any remaining step is cached as long as no rollback is triggered at that step.
- **Rolled-back step.** If the rollback indicator fires, $r_t = 1$, the step is converted into a full computation and opens a new segment.

On a rolled-back step, both the reference state and the error accumulation are reset before the process continues, $x_{\mathrm{ref}} := x_t$, $e_t := 0$, and $k \leftarrow k + 1$.

The parameters θ are fit offline on a calibration set of cached trajectories drawn from the same denoising pipeline, so that g_θ learns to score the drift of a cached step directly. Because the trajectories span diverse samples, timesteps, and cache schedules, we fit θ to the expected squared error over this population rather than to any single trajectory, which makes the estimator adapt to each sample and step at inference time,

$$\theta^\star \;=\; \arg\min_\theta \; \mathbb{E}_{t\sim\mathcal{S}}\Big[\, \big\|\, g_\theta(\mathbf{f}_t) - e_t \,\big\|_2^2 \Big], \qquad e_t \;=\; \frac{\big\|x_t^{\mathrm{gt}} - x_t^{\mathrm{pred}}\big\|_2}{\big\|x_t^{\mathrm{gt}}\big\|_2}, \tag{7}$$

where the expectation is taken over the cached steps t of the calibration segments $\mathcal{S}$, which are collected from fixed-schedule caching passes without error estimation, and the regression target e_t is the relative error of the cached latent with respect to its full-computation counterpart, with x_t^{pred} the latent produced by the cheap cache prediction and x_t^{gt} its full-computation counterpart at step t. The g_θ has far fewer parameters than the diffusion backbone and is trained offline. The parameters θ are then frozen for inference, so no gradient is ever propagated through the diffusion backbone.

3.3 Time-Aware Autoregressive Error Estimator

As illustrated in Figure 1(b), we instantiate the error score $\hat{e}_t$ used in the rollback loop with a lightweight autoregressive estimator g_θ. At each cached step t, the pipeline forms a compact, time-aware feature vector from the cached trajectory and its segment reference,

$$\mathbf{f}_t \;=\; \Big[\, \underbrace{\|x_t - x_{\mathrm{ref}}\|_2}_{\text{relative drift}},\; \underbrace{\Delta_t}_{\text{drifted time}},\; \underbrace{t_{\mathrm{ref}}}_{\text{segment phase}},\; \underbrace{e_{t-1}}_{\text{accumulated error}} \,\Big]^\top \in \mathbb{R}^4, \tag{8}$$

where the relative drift is the difference between the cached latent and its segment reference, t_{ref} encodes the reference denoising time, $\Delta_t = t_{\mathrm{ref}} - t_t$ measures the interval from the segment's

opening full computation, and e_{t-1} is the accumulated error carried over from the previous cached step. instead of scoring each step independently, the input directly expresses the history of drift accumulated along the denoising timeline, thus the prediction can depend on how far the segment has already deviated.

The map g_θ is realized by a small multilayer perceptron (MLP). It reads the features $\mathbf{f}_t$ of a single step and shares the same weights across all time steps, folding the sequence into the batch dimension and maps each $\mathbf{f}_t$ independently.

$$\hat{e}_t \;=\; \sum_i \big|\, \mathrm{MLP}_\theta(\mathbf{f}_t) \,\big|_i, \tag{9}$$

where the sum runs over the output units. We observe empirically that the accumulated error grows monotonically within a segment, so the absolute value simply enforces the learning of a non-negative score.

The compact estimator condenses the temporal latents of a segment into a handful of features and absorbs them into a shared mapping. Each cached step introduces only a fixed number of small matrix-vector products, whose cost is negligible relative to that of a full network forward pass.

4 Experiments

Evaluation and Metrics. We evaluate our method on two representative generation tasks and compare it against training-free acceleration baselines, including FORA Selvaraju et al. (2024), Δ-DiT Chen et al. (2024), ToCa Zou et al. (2025), TeaCache Liu et al. (2025a), TaylorSeer Liu et al. (2025b), DuCa Zou et al. (2026), and SpeCa Liu et al. (2025c), together with naive uniform step reduction. The first task is text-to-image generation with FLUX.1-dev, where we report the Image Reward Xu et al. (2023) and GenEval Ghosh et al. (2023) Overall scores together with the achieved acceleration. We measure every method against the same 50-step baseline and report the deviation from it. For video generation task, we quantify the performance with full-reference reconstruction metrics, i.e. PSNR, LPIPS Zhang et al. (2018), and SSIM Wang et al. (2004). In both settings we report the TFLOPs and the corresponding speedup to characterize the quality-efficiency trade-off.

4.1 Text-to-Image Generation

Table 1 reports a comparison on text-to-image generation with FLUX.1-dev, where every method is measured against the same 50-step reference baseline. The upper block lists naive uniform step reduction and Δ-DiT. The lower blocks list feature-caching methods such as FORA, ToCa and TaylorSeer, together with the cache-management schedulers TeaCache, DuCa and SpeCa. Most baselines suffer a severe drop.

The strongest baselines concentrate in the high-acceleration regime, which is precisely where our method operates. There other cache scheduling methods begin to collapse. Once the cache interval grows, the reused features drift from the true trajectory and the error accumulates unchecked. In contrast, our method degrades slowly and evenly as the acceleration grows. On image quality, ERAC attains $6.04\times$ FLOPs speedup at 0.9736 Image Reward, a deviation of only 0.0162 from the full baseline, and it still retains 0.9358 Image Reward at $7.08\times$. On the other hand, the baseline deviates by 0.0567 at $5.55\times$ and by 0.0543 at $6.34\times$ at the same speedups, which means ERAC reduces the quality loss by more than a factor of three while running faster. On prompt alignment, ERAC holds a Geneval Overall of 0.6367 at $6.04\times$, a deviation of 0.0385, whereas the strongest baseline falls to 0.6338 at $5.70\times$ and to 0.5922 at $6.34\times$. ERAC preserves the text-image correspondence better even though it accelerates more. These results suggest that aggressive acceleration can be achieved without sacrificing either image quality or prompt alignment.

4.2 Text-to-Video Generation

Table 2 reports a comparison on text-to-video generation with Wan2.1 1.3B, where every method is measured against the same 50-step reference baseline. The upper block lists moderate acceleration around $2\times$ and the lower block lists aggressive acceleration around $3\times$, covering feature-caching methods such as TeaCache and TaylorSeer together with the cache-management scheduler SeaCache. At roughly $2\times$, SeaCache reaches 26.60 PSNR at $2.08\times$, whereas our method attains

Table 1: Quantitative comparison in text-to-image generation for FLUX.

Method FLUX.1-dev	Efficient Attention	Acceleration		Image Reward ↑ DrawBench	Geneval ↑ Overall
		FLOPs(T) ↓	Speed ↑		
50 steps	✓	3719.50	1.00×	0.9898 (±0.0000)	0.6752 (±0.0000)
60% steps	✓	2231.70	1.67×	0.9739 (-0.0159)	0.6546 (-0.0206)
50% steps	✓	1859.75	2.00×	0.9429 (-0.0469)	0.6655 (-0.0097)
40% steps	✓	1487.80	2.62×	0.9317 (-0.0581)	0.6606 (-0.0146)
34% steps	✓	1264.63	3.13×	0.9346 (-0.0552)	0.6501 (-0.0251)
Δ-DiT ($N = 2$)	✓	2480.01	1.50×	0.9316 (-0.0582)	0.6700 (-0.0052)
Δ-DiT ($N = 3$)	✓	1686.76	2.21×	0.8561 (-0.1337)	0.6379 (-0.0373)
FORA ($N = 7$)	✓	670.44	5.55×	0.7398 (-0.2500)	0.5678 (-0.1074)
ToCa ($N = 10, N = 90\%$)	×	714.66	5.20×	0.7131 (-0.2767)	0.6026 (-0.0726)
DuCa ($N = 9, N = 90\%$)	✓	690.26	5.39×	0.8601 (-0.1297)	0.6189 (-0.0563)
TeaCache ($l = 1.2$)	✓	669.27	5.56×	0.7351 (-0.2547)	0.5845 (-0.0907)
TaylorSeer ($N = 7, O = 2$)	✓	670.44	5.55×	0.9331 (-0.0567)	0.5886 (-0.0866)
SpeCa	✓	652.14	5.70×	0.9717 (-0.0181)	0.6338 (-0.0414)
ERAC (Ours)	✓	**615.81**	**6.04×**	**0.9736 (-0.0162)**	**0.6367 (-0.0385)**
FORA ($N = 7$)	✓	596.07	6.24×	0.5442 (-0.4456)	0.5199 (-0.1553)
ToCa ($N = 10, N = 90\%$)	✓	644.70	5.77×	0.7131 (-0.2767)	0.5479 (-0.1273)
DuCa ($N = 9, N = 90\%$)	✓	606.91	6.13×	0.6753 (-0.3145)	0.5561 (-0.1191)
TeaCache ($l = 1.4$)	✓	594.90	6.25×	0.7346 (-0.2552)	0.5524 (-0.1228)
TaylorSeer ($N = 9, O = 2$)	✓	596.07	6.24×	0.8168 (-0.1730)	0.5380 (-0.1372)
SpeCa	✓	586.93	6.34×	0.9355 (-0.0543)	0.5922 (-0.0830)
ERAC (Ours)	✓	**525.35**	**7.08×**	**0.9358 (-0.0540)**	**0.6015 (-0.0737)**

Colored: ERAC attains the best quality among accelerated methods, with the smallest loss relative to the full-computation baseline.

Table 2: Quantitative comparison in text-to-video generation for Wan2.1 1.3B.

Method	TFLOPs ↓	Speed ↑	PSNR ↑	LPIPS ↓	SSIM ↑
Original (50 steps)	8214	1.00×	–	–	–
TeaCache ($\delta = 0.09$) Liu et al. (2025a)	4107	2.00×	20.84	0.171	0.721
TaylorSeer ($N = 2$) Liu et al. (2025b)	4189	1.96×	16.15	0.336	0.543
SeaCache ($\delta = 0.2$) Chung et al. (2026)	3942	2.08×	26.60	0.075	0.873
ERAC (Ours)	**3838**	**2.14×**	**27.12**	**0.064**	**0.902**
TeaCache ($\delta = 0.15$) Liu et al. (2025a)	2957	2.78×	18.88	0.245	0.645
TaylorSeer ($N = 3$) Liu et al. (2025b)	2956	2.78×	14.18	0.455	0.453
SeaCache ($\delta = 0.35$) Chung et al. (2026)	2793	2.94×	21.78	0.170	0.740
ERAC (Ours)	**2702**	**3.04×**	**22.07**	**0.128**	**0.834**

27.12 PSNR at 2.14×, with lower LPIPS and higher SSIM. At roughly 3×, most baselines degrade sharply. TaylorSeer falls to 14.18 PSNR at 2.78× and TeaCache falls to 18.88 PSNR at 2.78×, while SeaCache reaches 21.78 PSNR at 2.94×. Our method stays ahead with 22.07 PSNR at 3.04×, a deviation of only 0.91 PSNR from the 2× baseline setting, and it preserves structure better with an SSIM of 0.834 against 0.740 for SeaCache. These results suggest that our method sustains visual fidelity in video generation even when the acceleration grows.

4.3 Qualitative Comparison

While the quantitative results in Table 2 and Table 1 characterize the quality–efficiency trade-off numerically, we further provide a qualitative comparison to illustrate how the proposed error-regulated caching behaves visually under aggressive acceleration. Figure 2 shows generated samples from FLUX.1-dev under our method and representative cache scheduling baselines at comparable speedups, together with the full-computation reference. TaylorSeer shows a clear decline in color saturation and illumination, which makes the output appear washed out. This decline also influence the shape of object and the alignment with prompts. SpeCa alters the style and the shape of the



Figure 2: Qualitative comparison of text-to-image generation at aggressive acceleration ratios on FLUX.1-dev. From left to right: the full-computation reference (Flux), TaylorSeer, SpeCa, and our ERAC. TaylorSeer suffers from noticeable attenuation in color and illumination, SpeCa distorts the style and shape of the generated content, whereas our method preserves high visual quality and remains more faithful to the prompt.

generated content, which could affect the detail meaning of users' imagination. Our method, by contrast, retains rich and stable visual quality and keeps a close correspondence with the original model and text prompt.

5 Conclusion

We presented ERAC, a training-free acceleration framework that treats cache reuse as a predict-and-rollback error-feedback loop. Instead of relying on a fixed cache interval, ERAC monitors the accumulated drift of each cached step and triggers a full computation only when the estimated error passes a budget threshold. The central component is a time-aware autoregressive error estimator that reads a compact four-dimensional feature of the cached step and its segment reference, and whose cost stays far below a full forward pass. The estimator is fit offline and frozen at inference, so the diffusion backbone is never retrained. Experiments on text-to-image and text-to-video generation support this design. As the acceleration grows, the quality loss of ERAC grows slowly and evenly, while the baselines degrade sharply in the same regime, which shows that direct error estimation yields a better quality and efficiency trade-off than fixed cache schedules.

Several directions remain open. The estimator is fit on a calibration set from one pipeline, so extending it to new models and new schedulers without refitting is a natural next step. Combining the rollback rule with token-wise or block-wise reuse may push the speedup further while preserving the same error control. We hope that direct error estimation offers a practical route to reliable and aggressive diffusion inference.

AI use statement

In this work, we used generative AI tools for language polishing of the manuscript. We have not used generative AI tools for generating scientific content, research ideas, or experimental results, and the analysis of results are not applicable to this work. Additionally, we used generative AI tools for improving the clarity and grammar of the text. We have reviewed all AI-assisted work. The

authors wrote all scientific content and verified all text after polishing. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Our method is built on open-source code and public models and datasets. The diffusion pipelines are implemented with the publicly released FLUX.1-dev and Wan2.1 1.3B checkpoints. All baselines are reproduced from their official open-source implementations with the recommended settings, and all evaluation metrics follow their public code. The calibration set is sampled from the same public pipelines.

References

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

Jiwoo Chung, Sangeek Hyun, MinKyu Lee, Byeongju Han, Geonho Cha, Dongyoon Wee, Youngjun Hong, and Jae-Pil Heo. Seacache: Spectral-evolution-aware cache for accelerating diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14283–14294, 2026.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 12606–12633, 2024.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

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

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

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

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363. IEEE, 2025a.

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

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. Speca: Accelerating diffusion transformers with speculative feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, pp. 10024–10033, New York, NY, USA, 2025c. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755331. URL https://doi.org/10.1145/3746027.3755331.

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, 2024.

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.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

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.

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

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 *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Qixi Zheng, Yushen Chen, Zhikang Niu, Ziyang Ma, Xiaofei Wang, Kai Yu, and Xie Chen. Accelerating flow-matching-based text-to-speech via empirically pruned step sampling. In *Interspeech 2025*, pp. 2445–2449, 2025. doi: 10.21437/Interspeech.2025-2449.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 75467–75488, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/bbe024e0517fe12ac3a8d388b19ff9fe-Paper-Conference.pdf.

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. doi: 10.1109/TIP.2026.3698363.