ARCS: APPROXIMATION-RISK-AWARE COMPUTATION SCHEDULING FOR FEATURE FORECASTING IN DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion Transformers achieve strong generation quality but require expensive network evaluations across many denoising timesteps. Feature caching reduces this cost by reusing intermediate activations at subsequent timesteps. Recent methods mathematically approximate the temporal evolution of cached features to forecast their values, reducing errors associated with direct reuse. However, these forecasting methods often rely on predefined computation schedules, implicitly assuming that forecasts remain reliable for a predetermined duration. In practice, forecasting reliability varies substantially throughout denoising, making **when to compute** as important as **how much to compute**. We propose **Approximation-Risk-Aware Computation Scheduling (ARCS)**, a training-free method that adaptively allocates full computations according to estimated forecasting risk. We instantiate ARCS for Taylor feature forecasting, where adaptive computation produces irregularly spaced computation anchors. To handle these irregular anchors, we formulate Taylor forecasting with Newton divided differences and derive non-uniform approximation-error proxies that provide inexpensive online risk signals for determining when full computation is needed. Across image and video generation models, ARCS improves generation quality over fixed-interval Taylor forecasting, with the largest gains observed under aggressive DiT acceleration. In particular, ARCS reduces FID and sFID by up to 18.2% and 19.8%, respectively, compared with fixed-interval Taylor forecasting at matched full-computation budgets.

1 INTRODUCTION

Diffusion Transformers (DiTs) have become a powerful backbone for high-quality visual generation, but their inference remains computationally expensive because the denoising network must be repeatedly evaluated across many timesteps (Ho et al., 2020; Peebles & Xie, 2023). Feature caching reduces this cost by reusing intermediate activations across timesteps (Ma et al., 2024; Selvaraju et al., 2024; Zou et al., 2025). Recent forecasting-based approaches extend this idea by explicitly approximating the temporal trajectory of cached features to predict their values at subsequent timesteps (Liu et al., 2025b).

However, effective forecasting depends not only on *how* features are approximated, but also on *when* those approximations should be trusted. In forecasting methods with predefined computation schedules, the duration for which each forecast is used is determined in advance. In practice, forecasting reliability varies substantially throughout denoising.

As illustrated in figure 1(a), fixed scheduling allocates full computations independently of the varying forecasting risk, whereas risk-aware scheduling can place full computations when forecasts become unreliable. Thus, the number of full computations alone does not determine generation quality: **when to compute is as important as how much to compute**. This variability motivates estimating forecasting risk to guide the allocation of a limited full-computation budget.

This issue becomes more important under aggressive acceleration, where fewer full computations must cover longer forecasting spans and there are fewer opportunities to correct inaccurate predictions. This motivates our central question: **when should a feature forecast be trusted, and when**

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.

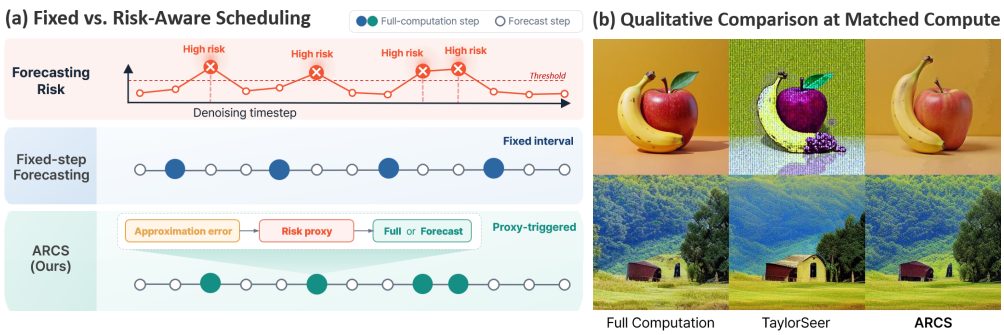

Figure 1: **Overview of ARCS.** (a) Conceptual comparison between fixed-interval feature forecasting and risk-aware scheduling in ARCS. Fixed-interval methods perform Full computations according to a predefined schedule, whereas ARCS estimates forecasting risk online and adaptively determines when a Full computation is needed. (b) Qualitative comparisons under aggressive acceleration on FLUX.1-dev (top, approximately 7 NFC) and DiT-XL/2 (bottom, 6 NFC), where NFC denotes the number of full computations. At computation budgets matched within each model, ARCS better preserves the structure and visual content of the Full-computation outputs.

should full computation be performed instead? Existing adaptive caching methods augment feature forecasters with auxiliary reliability checks (Guan et al., 2025; Liu et al., 2025c; Li et al., 2026). We instead use approximation-error analysis to estimate forecasting risk.

To this end, we propose **Approximation-Risk-Aware Computation Scheduling (ARCS)**, a training-free method that adaptively allocates full computations according to estimated forecasting risk. ARCS derives inexpensive online risk signals from the approximation-error structure and uses them to determine when full computation is needed. We instantiate ARCS for Taylor feature forecasting, where an approximation-error-based risk proxy directly guides the Full/Forecast decision. As illustrated in figure 1(a), this risk proxy enables online scheduling without additional network evaluations for risk estimation.

Adaptive computation produces irregularly spaced computation anchors, for which the uniform-spacing formulation of fixed-interval Taylor forecasting no longer applies directly. We therefore use Newton divided differences to formulate Taylor forecasting on non-uniform anchors and derive the risk proxy from the corresponding approximation-error analysis.

Figure 1(b) compares full computation, fixed-interval TaylorSeer, and ARCS on DiT and FLUX under aggressive acceleration. While TaylorSeer shows visible degradation, ARCS better preserves the global structure and fine details of the full-computation outputs.

Our main contributions are summarized as follows:

- We introduce ARCS, a training-free method that uses an approximation-error-derived risk proxy to guide adaptive Full/Forecast scheduling without additional network evaluations for risk estimation.
- We instantiate ARCS for Taylor feature forecasting under adaptively generated, irregular computation anchors. We use Newton divided differences to formulate the predictor and derive a risk proxy from an approximation-error analysis that accounts for non-uniform anchor spacing.
- Experiments across image and video generation models demonstrate improved generation quality over fixed-interval Taylor forecasting. In particular, under aggressive DiT acceleration, ARCS reduces FID and sFID by up to 18.2% and 19.8%, respectively, at matched full-computation budgets.

2 Related Work

Diffusion Acceleration and Feature Caching. Diffusion models have progressed from convolutional U-Net denoisers to scalable Diffusion Transformers (DiTs), enabling high-quality generation at the cost of expensive repeated network evaluations (Ho et al., 2020; Peebles & Xie, 2023). Existing acceleration methods either reduce the number of sampling steps through faster solvers or distillation (Song et al., 2021; Lu et al., 2022; Salimans & Ho, 2022), or reduce the cost of each denoising step. Among the latter, feature caching exploits temporal redundancy by reusing intermediate activations across nearby timesteps. DeepCache and FORA reuse cached features at selected network components or timesteps (Ma et al., 2024; Selvaraju et al., 2024), while ToCa further considers token-wise differences in cache sensitivity (Zou et al., 2025). Although effective, direct reuse can become inaccurate when feature trajectories change substantially over longer intervals.

Mathematical Feature Forecasting. Feature forecasting models temporal feature evolution instead of directly reusing stale activations. TaylorSeer approximates higher-order feature derivatives using finite differences and forecasts future features through local Taylor expansions (Liu et al., 2025b). HiCache retains the Taylor-style cache-then-forecast structure but replaces the standard polynomial basis with scaled Hermite polynomials to better capture non-monotonic feature trajectories and improve numerical stability (Feng et al., 2026). Spectrum instead fits global Chebyshev polynomials to cached features using ridge regression, enabling forecasting over longer timestep ranges (Han et al., 2026). These methods demonstrate that diffusion feature trajectories can be represented using structured mathematical approximations. We instantiate ARCS for Taylor feature forecasting and focus on a complementary question: rather than introducing another predictor, we use estimated forecasting risk to determine when full computation is needed.

Adaptive Cache Scheduling. Beyond designing how features are reused or forecasted, an important remaining question is when full computation should be performed. TeaCache estimates output variation from timestep-modulated input differences (Liu et al., 2025a), while RFC uses input–output relations to estimate feature changes and prediction errors (Son et al., 2026). DPCache instead constructs path-dependent costs from a calibration set and applies dynamic programming to optimize an input-independent schedule under its defined cost objective (Cui et al., 2026). CG-Taylor and SpeCa use partially computed features or verification signals to assess prediction reliability (Guan et al., 2025; Liu et al., 2025c). In contrast, ARCS derives its scheduling signal from the approximation-error structure of the underlying mathematical predictor and uses the resulting risk estimate to adaptively place full computations.

3 Method

3.1 Preliminaries

Denoising process. Diffusion models generate samples through an iterative denoising process that progressively transforms an initial noisy state into a generated sample, requiring repeated evaluations of the denoising network across a sequence of timesteps. Let x_t denote the noisy sample at timestep t, and let the reverse denoising transition be written abstractly as

$$x_{t-1} = \mathcal{S}(x_t, f_\theta(x_t, t), t)\,, \tag{1}$$

where f_θ denotes the denoising network and $\mathcal{S}$ denotes the sampler update rule. Because f_θ is repeatedly evaluated throughout denoising, feature forecasting reduces computation by predicting selected intermediate features instead of recomputing them from scratch.

Let $F(t)$ denote an intermediate feature within $f_\theta(x_t, t)$ at denoising timestep t. We refer to timesteps at which the corresponding network computation is explicitly evaluated as *full-computation anchors*, and to timesteps whose features are predicted from previously cached information as *forecast steps*.

Taylor feature forecasting. Among existing feature forecasting methods, we take TaylorSeer (Liu et al., 2025b) as our baseline forecaster. Let τ_0 denote the most recent anchor and define $k = t - \tau_0$ as the signed displacement to a target forecast timestep t. As denoising proceeds toward smaller timestep indices, $k < 0$ for subsequent forecast steps, while $|k|$ denotes the prediction distance.

An m-th order Taylor predictor is written as

$$\widehat{F}_m(t) = \sum_{r=0}^{m} \frac{D_r(\tau_0)}{r!} k^r, \tag{2}$$

where $D_r(\tau_0)$ denotes an approximation of the r-th temporal derivative of the feature trajectory, with $D_0(\tau_0) = F(\tau_0)$.

3.2 ARCS Overview

Fixed-interval forecasting determines the duration of each forecast without considering how reliable the current prediction is. ARCS instead makes the computation schedule depend on the estimated risk of the current feature forecast. ARCS derives its risk proxy from an approximation-error bound for the underlying forecaster. At each denoising timestep, ARCS evaluates inexpensive risk estimates and uses them to decide whether to perform a full network computation or continue forecasting.

ARCS combines the module-level risk estimates into a single timestep-level score $P_m(t)$ for the currently available prediction order m. Let $a_t \in \{\text{Full}, \text{Forecast}\}$ denote the computation decision at timestep t. Given a time-dependent decision threshold $\gamma(t)$, ARCS applies

$$a_t = \begin{cases} \text{Full}, & P_m(t) > \gamma(t), \\ \text{Forecast}, & P_m(t) \le \gamma(t). \end{cases} \tag{3}$$

During initialization, the available anchor history may be insufficient to construct both a higher-order predictor and its corresponding risk proxy, as the proxy requires an additional anchor to estimate the next-order variation. ARCS therefore uses the highest effective order for which both the predictor and its corresponding risk proxy can be constructed, progressively increasing the order as additional full-computation anchors become available.

ARCS makes one shared Full/Forecast decision at each denoising timestep. When a Full computation is triggered, the network features are recomputed and stored as a new full-computation anchor, updating the cached history used for subsequent forecasts. Otherwise, the predicted features are used without introducing a new anchor.

Because Full decisions depend on the evolving forecasting risk, the resulting computation anchors are generally irregularly spaced. The predictor must therefore account for the actual anchor locations rather than assume a fixed computation interval. In section 3.3, we formulate Taylor forecasting over irregular anchors using Newton divided differences. We then derive the approximation-error-based risk proxy and adaptive scheduling rule used by ARCS in section 3.4.

3.3 Taylor Forecasting under Irregular Anchors

In fixed-interval Taylor forecasting, equally spaced computation anchors allow temporal derivatives to be estimated with standard finite differences, as in TaylorSeer (Liu et al., 2025b). For example, with interval N, the second-order estimate is

$$D_2(\tau_0) = \frac{F(\tau_0) - 2F(\tau_0 + N) + F(\tau_0 + 2N)}{N^2}. \tag{4}$$

ARCS selects Full steps adaptively, so its computation anchors need not be equally spaced. Let $\tau_0, \ldots, \tau_m$ denote the most recent $m+1$ anchors, with τ_0 being the most recent, and let $h_i = |\tau_{i-1} - \tau_i|$ denote their consecutive spacings. We extend the fixed-interval Taylor forecasting formulation to irregularly spaced computation anchors using Newton divided differences, estimating the r-th temporal derivative as

$$D_r(\tau_0) = r!\, F[\tau_0, \ldots, \tau_r]. \tag{5}$$

For the second-order predictor, this gives

$$D_2(\tau_0) = \frac{2}{\tau_0 - \tau_2} \left(\frac{F(\tau_0) - F(\tau_1)}{\tau_0 - \tau_1} - \frac{F(\tau_1) - F(\tau_2)}{\tau_1 - \tau_2} \right). \tag{6}$$

Substituting these estimates into equation 2 yields the forecast for irregular anchors. When the anchor spacings are equal, the second-order expression reduces to the fixed-interval estimate above.

The divided-difference definition, derivation, and fixed-spacing special case are provided in Appendix A.1; we analyze the spacing-dependent approximation error in Appendix A.2.

3.4 Approximation-Error Proxy and Adaptive Scheduling

Taylor-derived forecasting risk. The error of an m-th order Taylor forecast consists of errors in the derivatives estimated from computation anchors and the truncation error of the Taylor expansion. For a forecast timestep t, this decomposition gives

$$\left\| F(t) - \widehat{F}_m(t) \right\|_2 \leq \mathcal{E}_{\text{deriv}}(t) + \frac{|k|^{m+1}}{(m+1)!} M_{m+1}, \tag{7}$$

where $k = t - \tau_0$, $\mathcal{E}_{\text{deriv}}(t)$ collects the spacing-dependent derivative-approximation terms, and M_{m+1} bounds the magnitude of the next-order derivative over the relevant timestep interval. ARCS uses the truncation component as its online risk signal. The full error-bound derivation and proxy variants that also include derivative-approximation terms are provided in Appendices A.2 and A.4, respectively.

The next-order derivative magnitude is unavailable at forecast steps. For each forecasted module ℓ, we estimate its local variation from cached full-computation anchors:

$$\widehat{M}_{m+1,\ell} = \left\| (m+1)!\, F_\ell[\tau_0, \ldots, \tau_{m+1}] \right\|_2 . \tag{8}$$

This uses one more anchor than the $m+1$ anchors required by the m-th order predictor. The resulting module-level proxy is

$$p_{m,\ell}(t) = \frac{|k|^{m+1}}{(m+1)!} \widehat{M}_{m+1,\ell}. \tag{9}$$

Because $\widehat{M}_{m+1,\ell}$ is a local estimate rather than a guaranteed derivative bound, the proxy is a scheduling signal, not a certified upper bound on forecasting error.

Full/Forecast scheduling. ARCS uses the highest prediction order for which both the predictor and its risk proxy are available. It normalizes scores across effective orders and combines module-level proxies into the timestep-level score $P_m(t)$ used in equation 3. Initialization, normalization, and module aggregation are detailed in Appendix A.5.

Timestep-dependent confidence thresholds that are more permissive at later steps have also been explored (Guan et al., 2025). Motivated by this observation, we gradually relax the decision threshold over the sampling trajectory:

$$\gamma(t) = \gamma_0 \left[1 + (r_\gamma - 1) u_t^2 \right], \tag{10}$$

where $u_t \in [0, 1]$ increases from the beginning to the end of sampling. We use $r_\gamma = 1.15$, which makes the threshold modestly more permissive toward later timesteps. For each target full-computation budget, γ_0 is selected on the calibration set and fixed during evaluation. We evaluate the sensitivity to r_γ in Appendix B.1.

4 Experiments

4.1 Experimental Setup

Models and datasets. We evaluate ARCS on three representative diffusion transformer architectures. For class-conditional image generation, we use DiT-XL/2 with DDIM-50 sampling on ImageNet at 256×256 resolution. For text-to-image generation, we use FLUX.1-dev on the DrawBench prompt set at 992×992 resolution. For text-to-video generation, we use HunyuanVideo with 50-step sampling on the official VBench prompt set, generating 65-frame videos at 480×640 resolution.

Baselines. For DiT-XL/2, we compare ARCS with full-computation DDIM (Song et al., 2021), ToCa (Zou et al., 2025), TaylorSeer (Liu et al., 2025b), and adaptive forecasting methods SpeCa (Liu et al., 2025c) and CG-Taylor (Guan et al., 2025). For FLUX.1-dev, we compare against full computation, TaylorSeer, SpeCa, and CG-Taylor. For HunyuanVideo, we compare against full computation, TaylorSeer, and SpeCa. At each operating point, we configure the baselines and ARCS to use approximately matched full-computation budgets and report the realized number of full computations (NFC).

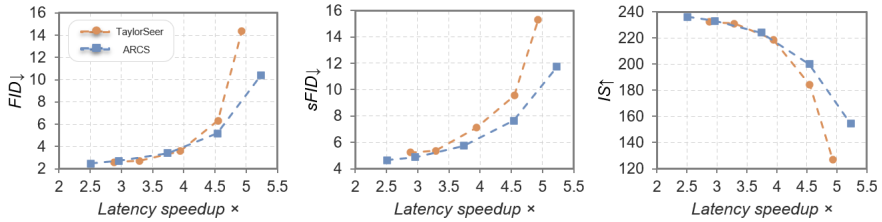

Figure 2: **Quality–latency trade-off on DiT-XL/2.** Comparison of ARCS and fixed-interval Taylor forecasting in terms of FID, sFID, and IS across different latency speedups relative to full computation.

Table 1: Quantitative comparison of ARCS and existing acceleration methods on DiT-XL/2 under three acceleration regimes with approximately matched full-computation budgets. TaylorSeer uses a maximum prediction order of $m = 2$. The comparison includes realized NFC, sampling latency, FLOPs, FID, sFID, and IS at each operating point.

Method	**NFC**	**Latency (s)** ↓	**FLOPs (T)** ↓	**FID** ↓	**sFID** ↓	**IS** ↑
Full computation	50.0	0.3604	23.74	2.26	4.28	239.33
ToCa	11.0	0.1153	6.25	8.54	8.68	178.37
TaylorSeer ($N = 5$)	11.0	0.1095	5.24	2.71	5.37	231.04
SpeCa	11.0	0.1087	5.41	**2.69**	5.21	<u>231.34</u>
CG-Taylor	11.0	0.0869	5.88	4.42	**4.82**	202.55
ARCS (Ours)	11.0	0.1216	5.25	<u>2.70</u>	<u>4.88</u>	**233.14**
ToCa	8.0	0.0921	4.69	59.93	44.54	46.77
TaylorSeer ($N = 7$)	8.0	0.0914	3.82	3.59	7.12	<u>218.69</u>
SpeCa	8.6	0.0922	4.42	**3.36**	<u>6.35</u>	218.41
CG-Taylor	8.0	0.0655	4.51	17.68	11.46	124.31
ARCS (Ours)	8.0	0.0963	3.82	<u>3.40</u>	**5.75**	**224.08**
ToCa	6.0	0.0869	4.32	104.76	64.27	19.23
TaylorSeer ($N = 10$)	6.0	0.0791	2.87	<u>6.32</u>	9.54	<u>184.25</u>
SpeCa	5.9	0.0748	3.32	8.55	<u>8.46</u>	153.34
CG-Taylor	6.0	0.0531	3.59	73.45	32.18	37.83
ARCS (Ours)	6.0	0.0794	2.87	**5.17**	**7.65**	**200.09**

Evaluation protocol and metrics. For DiT-XL/2, we report FID, sFID, and Inception Score (IS). For FLUX.1-dev, we report ImageReward and CLIP Score for text–image generation quality, together with PSNR, SSIM, and LPIPS for fidelity to the corresponding full-computation outputs. For HunyuanVideo, we report VBench for overall video generation quality, together with PSNR, SSIM, and LPIPS for fidelity to the corresponding full-computation outputs. We report sampling-only latency, including predictor computation and online scheduling overhead. Further implementation, calibration, and evaluation details are provided in Appendix C.

4.2 Main Results

Class-conditional image generation. Table 1 compares ARCS with existing acceleration methods on DiT-XL/2 at approximately matched computation budgets. At 8 NFC, ARCS achieves the best sFID and IS among the compared methods, while its FID remains comparable to the best-performing SpeCa (3.40 vs. 3.36). Compared with TaylorSeer at the same budget, ARCS reduces FID from 3.59 to 3.40 and sFID from 7.12 to 5.75, while increasing IS from 218.69 to 224.08. At the more aggressive 6-NFC regime, ARCS achieves the best FID, sFID, and IS among all compared acceleration methods. In particular, compared with TaylorSeer, ARCS reduces FID and sFID by 18.2% and 19.8%, respectively, and increases IS by 8.6%.

Table 2: Quantitative comparison of ARCS and existing acceleration methods on FLUX.1-dev and HunyuanVideo under two acceleration regimes with approximately matched full-computation budgets. TaylorSeer uses a maximum prediction order of $m = 1$.

Model	Method	NFC	Latency (s) ↓	FLOPs (T) ↓	ImageReward ↑	CLIP ↑	PSNR ↑	SSIM ↑	LPIPS ↓
FLUX.1-dev	Full computation	50.0	20.66	3477.45	0.9662	0.2776	–	–	–
	TaylorSeer ($N = 6$)	10.0	4.86	696.39	0.9148	0.2777	18.4418	0.6636	0.4147
	SpeCa	10.2	4.83	789.09	0.9194	0.2749	17.6234	0.6238	0.4597
	CG-Taylor	10.0	7.49	744.93	0.7601	0.2760	19.1690	0.6398	0.4452
	ARCS (Ours)	10.1	4.83	696.39	**0.9406**	**0.2802**	**20.3340**	**0.7083**	**0.3168**
	TaylorSeer ($N = 9$)	7.0	4.06	557.33	0.7163	0.2730	16.9450	0.5999	0.5229
	SpeCa	7.2	3.83	524.40	0.6729	0.2718	16.0976	0.4805	0.6441
	CG-Taylor	7.0	5.65	539.99	-1.0849	0.2418	14.3163	0.2897	0.8477
	ARCS (Ours)	7.1	3.59	487.81	**0.8450**	**0.2797**	**18.0010**	**0.6162**	**0.4424**
Model	**Method**	**NFC**	**Latency (s) ↓**	**FLOPs (T) ↓**	**VBench ↑**		**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
HunyuanVideo	Full computation	50.0	132.24	7520.00	80.60		–	–	–
	TaylorSeer ($N = 6$)	9.0	27.19	1359.19	79.60		16.2448	**0.7250**	0.2952
	SpeCa	9.0	29.23	1361.63	69.44		13.0488	0.5297	0.5764
	ARCS (Ours)	9.1	27.43	1377.34	**79.69**		**17.9652**	0.7155	**0.2569**
	TaylorSeer ($N = 9$)	6.0	19.42	908.40	**77.69**		15.4310	**0.6995**	**0.3368**
	SpeCa	6.4	22.46	960.93	66.24		12.6160	0.5286	0.6169
	ARCS (Ours)	6.1	18.96	922.78	77.44		**16.1437**	0.6529	0.3394

To examine the quality–latency trade-off over a wider range of operating points, figure 2 compares ARCS with fixed-interval Taylor forecasting across different latency speedups. As acceleration becomes more aggressive, fixed-interval forecasting exhibits a sharper degradation in generation quality, whereas ARCS maintains lower FID and sFID and higher IS in the high-speedup regime. Overall, the table and quality–latency curves show that the advantage of ARCS becomes more pronounced under more aggressive acceleration.

Text-to-image generation. Table 2 compares ARCS with TaylorSeer, SpeCa, and CG-Taylor on FLUX.1-dev at approximately matched computation budgets. At approximately 10 NFC, ARCS achieves the best performance across all generation-quality and full-output fidelity metrics among the compared acceleration methods.

The differences become more pronounced at approximately 7 NFC, where ARCS again achieves the best performance across all five quality and fidelity metrics. Compared with TaylorSeer, ARCS improves ImageReward by 18.0% and reduces LPIPS by 15.4%, while increasing CLIP Score from 0.2730 to 0.2797 and PSNR from 16.9450 to 18.0010. ARCS also substantially outperforms SpeCa and CG-Taylor in generation quality at this more aggressive operating point. Overall, these results show that ARCS more effectively preserves generation quality and fidelity to the full-computation outputs as acceleration becomes more aggressive.

Text-to-video generation. Table 2 compares ARCS with TaylorSeer and SpeCa on HunyuanVideo at approximately matched computation budgets. At approximately 9 NFC, ARCS achieves the best VBench, PSNR, and LPIPS among the compared acceleration methods. Compared with TaylorSeer, ARCS improves PSNR by 1.72 dB and reduces LPIPS by 13.0%, while maintaining a comparable VBench score (79.69 vs. 79.60).

At approximately 6 NFC, ARCS achieves the best PSNR, increasing it from 15.4310 to 16.1437 compared with TaylorSeer, while maintaining a comparable VBench score (77.44 vs. 77.69). Across both operating points, ARCS substantially outperforms SpeCa in VBench and full-output fidelity. Overall, these results show that ARCS remains competitive with TaylorSeer in overall video quality while providing stronger reconstruction fidelity, particularly in PSNR.

Qualitative results. Figure 3 provides qualitative comparisons on DiT-XL/2, FLUX.1-dev, and HunyuanVideo. Across all three models, ARCS better preserves the visual content and structure of the Full-computation outputs than fixed-interval Taylor forecasting under aggressive acceleration.

4.3 Error-Proxy Ablation

Motivated by the approximation-error bound derived in Section 3.4, we consider three candidate risk proxies corresponding to different components of the bound. The **Taylor truncation proxy**, used by the final ARCS scheduler, captures the Taylor truncation component of the bound. The **divided-**



Figure 3: Qualitative comparison of Full computation, TaylorSeer, and ARCS. Left: HunyuanVideo with the prompt "A panda drinking coffee in a cafe in Paris, oil painting" (NFC: 50/9/9). Right: DiT-XL/2 with the goldfish class (NFC: 50/6/6) and FLUX.1-dev with DrawBench prompt P193 (NFC: 50/7/7). ARCS better preserves the visual content and structure of the Full-computation outputs under aggressive acceleration.

Table 3: Ablation of error proxies on DiT-XL/2 at approximately matched computation budgets with an average NFC of 11. The Taylor truncation proxy estimates forecasting risk from the truncation behavior of the Taylor predictor, while the other rows evaluate alternative proxy formulations based on the full expansion and divided differences.

Error Proxy	Latency (s) ↓	FLOPs (T) ↓	FID ↓	sFID ↓	IS ↑
Taylor truncation (Ours)	0.1192	5.2392	**2.70**	**4.88**	**233.14**
Full-term proxy	0.1196	5.2392	3.84	5.29	224.83
Divided-difference proxy	0.1232	5.2392	13.28	38.81	144.53

difference proxy uses the divided-difference component, while the **full-term proxy** combines both components. Detailed formulations are provided in the Appendix.

As shown in Table 3, the Taylor truncation proxy achieves the best generation quality across FID, sFID, and IS under nearly identical computation budgets. The alternative proxy variants lead to larger quality degradation, supporting the Taylor truncation proxy as the primary scheduling signal used by ARCS.

4.4 Risk Proxy–Forecasting Error Correlation

We examine whether the proposed risk proxy reflects the actual error introduced by Taylor forecasting throughout the denoising trajectory. We run ARCS with its final scheduling configuration and, at every proxy-valid timestep, compute both the Taylor-forecasted module features and their corresponding Full-computation features. Their difference provides the oracle forecasting error. These additional Full computations are used only for analysis and do not affect the scheduling decisions, cache updates, or generated samples. Timesteps at which the required prediction history is unavailable are excluded.

For each module, we compute the relative forecasting error between the Taylor-forecasted feature and its corresponding Full-computation feature. Module-level forecasting errors are aggregated using RMS, consistently with the timestep-level aggregation used by ARCS. We then compare the resulting oracle forecasting error with the Taylor truncation proxy across denoising timesteps.

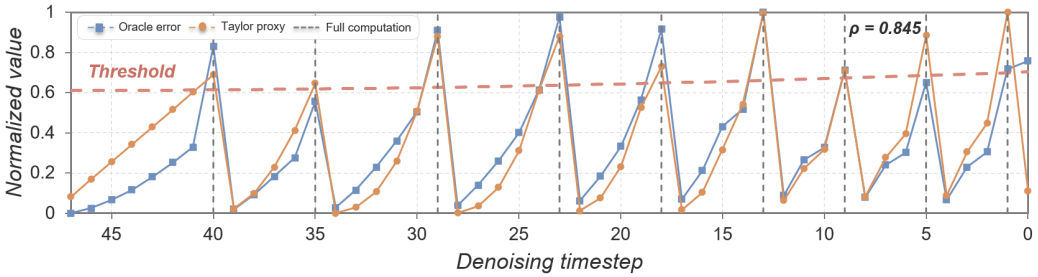

Figure 4: Risk proxy and oracle forecasting error over the DiT-XL/2 denoising trajectory. The Taylor truncation proxy closely follows the temporal variation of the oracle error. Gray vertical lines indicate Full-computation timesteps, and the red dashed line denotes the adaptive decision threshold.

As shown in figure 4, the Taylor truncation proxy closely tracks the temporal variation of the oracle forecasting error, yielding a Spearman rank correlation of $\rho = 0.845$ over all proxy-valid timesteps. This result shows that the proxy follows the overall variation of the oracle error throughout the denoising trajectory. Between consecutive Full computations, both signals generally increase as forecasting continues, forming repeated rising patterns. When a Full computation refreshes the feature history, both the proxy and oracle error drop sharply before beginning to accumulate again. Although their exact magnitudes do not always coincide, their peaks and relative variations are well aligned across most of the trajectory.

The all-timestep analysis includes both Forecast-selected and Full-selected timesteps. We therefore additionally evaluate only the Forecast-selected timesteps, at which ARCS actually uses the predicted features. The correlation remains strong in this subset ($\rho = 0.741$). Thus, the all-timestep result shows that the proxy tracks the oracle error over the complete trajectory, while the forecast-only result confirms that the proxy remains informative in the regions where Taylor forecasting is actually applied.

These observations support the use of a risk proxy derived from the structure of the approximation-error bound for Taylor forecasting: rather than directly evaluating the true forecasting error, ARCS estimates its risk from cached feature histories and uses this signal to identify when forecasting becomes unreliable and a Full computation should be performed.

5 Conclusion

We presented ARCS, a training-free method that derives forecasting-risk estimates from approximation-error analysis to guide adaptive Full/Forecast scheduling. Taking TaylorSeer as our baseline forecaster, we extended fixed-interval Taylor forecasting to irregularly spaced computation anchors using Newton divided differences. The resulting risk proxy can be evaluated from cached features and their computation times without additional network evaluations. Correlation analyses and matched-budget proxy ablations support the Taylor truncation proxy as an effective signal for assessing forecasting reliability and guiding computation allocation. We evaluated ARCS on both image and video generation models. On DiT-XL/2, ARCS consistently improves generation quality over fixed-interval Taylor forecasting at matched full-computation budgets, with larger gains under aggressive acceleration. At 6 NFC, ARCS reduces FID and sFID by 18.2% and 19.8%, respectively, compared with TaylorSeer. Similar improvements are observed on FLUX.1-dev, where ARCS improves ImageReward by 18.0% and reduces LPIPS by 15.4% at approximately 7 NFC.

Limitations and future work. ARCS is currently instantiated with Taylor feature forecasting; extending it to other forecasters requires corresponding approximation-error proxies that can be estimated from cached histories. The current proxy estimates local forecasting risk without explicitly modeling how feature errors propagate through the remaining denoising process. Future work includes extending error-derived scheduling to other forecasting formulations, incorporating error propagation into risk estimation, and developing online threshold adaptation for explicit per-sample budget control.

AI USE STATEMENT

We used generative AI tools to discuss research ideas and experimental design, receive feedback on mathematical derivations and proof arguments, identify relevant literature, and draft, translate, and edit parts of the manuscript. The authors checked the mathematical claims, references, and experimental results and revised the AI-assisted text. The authors take responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

We describe the forecasting-risk proxy and scheduling procedure in Section 3.4. Derivations and scheduler details are provided in Appendix A.2 and Appendix A.5. Experimental configurations, calibration, and evaluation procedures are described in Appendix C.

REFERENCES

Black Forest Labs. FLUX.1-dev. Model card, 2024.

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

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.

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

Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, and Linfeng Zhang. HiCache: A plug-in scaled-hermite upgrade for taylor-style cache-then-forecast diffusion acceleration. In *International Conference on Learning Representations*, 2026.

Xiaoliu Guan, Lielin Jiang, Hanqi Chen, Xu Zhang, Jiaxing Yan, Guanzhong Wang, Yi Liu, Zetao Zhang, and Yu Wu. Forecasting when to forecast: Accelerating diffusion models with confidence-gated taylor. *Knowledge-Based Systems*, 330:114635, 2025. doi: 10.1016/j.knosys.2025.114635.

Jiaqi Han, Juntong Shi, Puheng Li, Haotian Ye, Qiushan Guo, and Stefano Ermon. Adaptive spectral feature forecasting for diffusion sampling acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

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

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

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, et al. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, et al. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Yanchao Li, Jiaqing Xie, Ben Gao, Wanhao Liu, Yanbo Wang, TY Tsui, Jinfei Liu, Yuqiang Li, and Tianfan Fu. Disagree to accelerate: Closing the loop on diffusion feature forecasts. *arXiv preprint arXiv:2608.01740*, 2026.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *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*, 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*, 2025c. doi: 10.1145/3746027.3755331.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5775–5787, 2022.

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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, 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. In *Advances in Neural Information Processing Systems*, 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.

Byunggwan Son, Jeimin Jeon, Jeongwoo Choi, and Bumsub Ham. Relational feature caching for accelerating diffusion transformers. In *International Conference on Learning Representations*, 2026.

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

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. In *Advances in Neural Information Processing Systems*, 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/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

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.

A Derivation of Taylor Error Proxies under Irregular Anchors

A.1 Taylor Forecasting under Irregular Computation Anchors

Taylor-style feature forecasting approximates the temporal evolution of an intermediate feature $F(t)$ using features obtained at previous full-computation steps. Under fixed-interval scheduling, full computations are performed at equally spaced timesteps, allowing the temporal derivatives required by the Taylor predictor to be approximated using standard finite differences, as in TaylorSeer (Liu et al., 2025b).

ARCS instead determines full-computation steps adaptively. As a result, the computation anchors are generally irregularly spaced, and the fixed-spacing finite-difference formulation no longer directly applies. We therefore formulate Taylor forecasting using Newton divided differences, which naturally accommodate arbitrary computation-anchor locations.

Let

$$\tau_0, \tau_1, \ldots, \tau_m$$

denote the most recent $m+1$ full-computation anchors, indexed by recency, where τ_0 is the most recent anchor and τ_i denotes the i-th preceding anchor. Under the timestep convention in section 3.1, denoising proceeds toward smaller timestep indices. Therefore, previously observed anchors have larger timestep indices:

$$\tau_0 < \tau_1 < \cdots < \tau_m.$$

We define the interval length between consecutive anchors as

$$h_i = |\tau_{i-1} - \tau_i| = \tau_i - \tau_{i-1}, \qquad i = 1, \ldots, m.$$

Under adaptive scheduling, these intervals need not be equal:

$$h_1, h_2, \ldots, h_m \quad \text{need not be equal.}$$

We assume that the feature trajectory is sufficiently smooth over the relevant timestep interval and that the computation anchors are distinct.

For arbitrary anchor locations, the first-order divided difference is defined as

$$F[\tau_i, \tau_j] = \frac{F(\tau_i) - F(\tau_j)}{\tau_i - \tau_j},$$

and higher-order divided differences are recursively defined by

$$F[\tau_0, \ldots, \tau_r] = \frac{F[\tau_0, \ldots, \tau_{r-1}] - F[\tau_1, \ldots, \tau_r]}{\tau_0 - \tau_r}.$$

We use the r-th divided difference to construct an approximation of the r-th temporal derivative associated with the most recent anchor:

$$D_r(\tau_0) = r!\, F[\tau_0, \ldots, \tau_r], \qquad r = 1, \ldots, m,$$

with

$$D_0(\tau_0) = F(\tau_0).$$

Let t denote a forecast timestep and define its signed offset from the most recent computation anchor as

$$k = t - \tau_0.$$

Because subsequent denoising steps have smaller timestep indices, $k < 0$ for a forecast step, while $|k|$ denotes the prediction distance. The resulting m-th order Taylor forecast is

$$\widehat{F}_m(t) = \sum_{r=0}^{m} \frac{D_r(\tau_0)}{r!} k^r.$$

When the computation anchors are equally spaced,

$$h_1 = h_2 = \cdots = h_m = N,$$

the divided-difference formulas become exactly the standard fixed-spacing finite-difference formulas used in TaylorSeer. Thus, this formulation recovers fixed-interval Taylor forecasting as a special case while remaining applicable to the irregular anchors produced by adaptive scheduling.

For example, consider a second-order predictor with three computation anchors

$$\tau_0, \qquad \tau_1, \qquad \tau_2.$$

The first-order derivative approximations over the two adjacent anchor intervals are

$$D_1^{\mathrm{cur}} = \frac{F(\tau_0) - F(\tau_1)}{\tau_0 - \tau_1},$$

and

$$D_1^{\mathrm{prev}} = \frac{F(\tau_1) - F(\tau_2)}{\tau_1 - \tau_2}.$$

The second-order derivative approximation is obtained from the second Newton divided difference:

$$D_2(\tau_0) = \frac{2}{\tau_0 - \tau_2} \left(D_1^{\mathrm{cur}} - D_1^{\mathrm{prev}} \right).$$

The corresponding second-order forecast is then

$$\widehat{F}_2(t) = F(\tau_0) + D_1^{\mathrm{cur}} k + \frac{1}{2} D_2(\tau_0) k^2.$$

To verify the fixed-spacing special case, let

$$\tau_1 = \tau_0 + N, \qquad \tau_2 = \tau_0 + 2N.$$

Then

$$D_1^{\mathrm{cur}} = \frac{F(\tau_1) - F(\tau_0)}{N}, \qquad D_1^{\mathrm{prev}} = \frac{F(\tau_2) - F(\tau_1)}{N}.$$

Since

$$\tau_0 - \tau_2 = -2N,$$

we obtain

$$D_2(\tau_0) = \frac{F(\tau_0) - 2F(\tau_1) + F(\tau_2)}{N^2},$$

which recovers the standard second-order fixed-spacing finite-difference derivative estimate used in fixed-interval Taylor forecasting.

The accuracy of $D_r(\tau_0)$ as an approximation of the true derivative $F^{(r)}(\tau_0)$ depends on the anchor spacings $h_1, \ldots, h_r$. We derive this spacing-dependent approximation error in the following subsection.

A.2 Derivation of the Irregular-Anchor Approximation Error

The Taylor forecast in equation 2 contains two distinct sources of approximation error. First, truncating the Taylor expansion at order m introduces a Taylor truncation error. Second, the temporal derivatives $F^{(r)}(\tau_0)$ are not directly available and are replaced by the divided-difference approximations $D_r(\tau_0)$, introducing an additional derivative-approximation error.

Let

$$k = t - \tau_0$$

denote the signed offset from the most recent computation anchor. The exact Taylor expansion around τ_0 can be written as

$$F(t) = \sum_{r=0}^{m} \frac{F^{(r)}(\tau_0)}{r!} k^r + R_{m+1}(t),$$

where the truncation remainder satisfies

$$\|R_{m+1}(t)\|_2 \leq \frac{|k|^{m+1}}{(m+1)!} M_{m+1},$$

with

$$M_{m+1} = \sup_{\xi \in \mathcal{I}_t} \left\| F^{(m+1)}(\xi) \right\|_2,$$

where $\mathcal{I}_t$ denotes the interval between t and τ_0.

Subtracting the forecast

$$\widehat{F}_m(t) = \sum_{r=0}^{m} \frac{D_r(\tau_0)}{r!} k^r$$

gives

$$F(t) - \widehat{F}_m(t) = \sum_{r=1}^{m} \frac{F^{(r)}(\tau_0) - D_r(\tau_0)}{r!} k^r + R_{m+1}(t).$$

Therefore,

$$\left\| F(t) - \widehat{F}_m(t) \right\|_2 \leq \sum_{r=1}^{m} \frac{|k|^r}{r!} \left\| F^{(r)}(\tau_0) - D_r(\tau_0) \right\|_2 + \frac{|k|^{m+1}}{(m+1)!} M_{m+1}.$$

This decomposition separates the error caused by approximating the temporal derivatives from the truncation error of the Taylor expansion. While the latter depends directly on the prediction distance $|k|$, the former additionally depends on the spacing of the computation anchors.

First-order derivative approximation. Consider the two most recent computation anchors τ_0 and τ_1, with

$$h_1 = |\tau_0 - \tau_1| = \tau_1 - \tau_0.$$

Under our timestep convention,

$$\tau_1 = \tau_0 + h_1.$$

A first-order Taylor expansion with remainder gives

$$F(\tau_1) = F(\tau_0) + h_1 F'(\tau_0) + R_2^{(1)},$$

where

$$\left\| R_2^{(1)} \right\|_2 \leq \frac{h_1^2}{2} M_2, \qquad M_2 = \sup_{\xi \in [\tau_0, \tau_1]} \left\| F''(\xi) \right\|_2.$$

The first-order divided-difference approximation is

$$D_1(\tau_0) = \frac{F(\tau_0) - F(\tau_1)}{\tau_0 - \tau_1} = \frac{F(\tau_1) - F(\tau_0)}{h_1}.$$

Substituting the expansion above yields

$$D_1(\tau_0) = F'(\tau_0) + \frac{R_2^{(1)}}{h_1}.$$

Taking the norm and using the remainder bound gives

$$\left\| F'(\tau_0) - D_1(\tau_0) \right\|_2 \leq C_{\text{bound}}^{(1)}(h_1) M_2,$$

where

$$C_{\text{bound}}^{(1)}(h_1) = \frac{h_1}{2}.$$

Second-order derivative approximation. We next consider three computation anchors

$$\tau_0, \qquad \tau_1, \qquad \tau_2,$$

with consecutive interval lengths

$$h_1 = \tau_1 - \tau_0, \qquad h_2 = \tau_2 - \tau_1.$$

Under the same timestep convention,

$$\tau_1 = \tau_0 + h_1, \qquad \tau_2 = \tau_0 + (h_1 + h_2).$$

Expanding the two preceding anchor features around τ_0 to second order gives

$$F(\tau_1) = F(\tau_0) + h_1 F'(\tau_0) + \frac{h_1^2}{2} F''(\tau_0) + R_3^{(1)},$$

and

$$F(\tau_2) = F(\tau_0) + (h_1 + h_2) F'(\tau_0) + \frac{(h_1 + h_2)^2}{2} F''(\tau_0) + R_3^{(2)},$$

where

$$\left\| R_3^{(1)} \right\|_2 \leq \frac{h_1^3}{6} M_3, \qquad \left\| R_3^{(2)} \right\|_2 \leq \frac{(h_1 + h_2)^3}{6} M_3,$$

and

$$M_3 = \sup_{\xi \in [\tau_0, \tau_2]} \left\| F'''(\xi) \right\|_2 .$$

The first divided difference over the most recent interval is

$$D_1^{\text{cur}} = \frac{F(\tau_0) - F(\tau_1)}{\tau_0 - \tau_1} = \frac{F(\tau_1) - F(\tau_0)}{h_1},$$

which can be written as

$$D_1^{\text{cur}} = F'(\tau_0) + \frac{h_1}{2} F''(\tau_0) + \frac{R_3^{(1)}}{h_1}.$$

Similarly, the first divided difference over the preceding interval is

$$D_1^{\text{prev}} = \frac{F(\tau_1) - F(\tau_2)}{\tau_1 - \tau_2} = \frac{F(\tau_2) - F(\tau_1)}{h_2},$$

and substitution of the Taylor expansions yields

$$D_1^{\text{prev}} = F'(\tau_0) + \left(h_1 + \frac{h_2}{2} \right) F''(\tau_0) + \frac{R_3^{(2)} - R_3^{(1)}}{h_2}.$$

Using these quantities, the second-order derivative approximation is

$$D_2(\tau_0) = \frac{2}{\tau_0 - \tau_2} \left(D_1^{\text{cur}} - D_1^{\text{prev}} \right).$$

Since

$$\tau_0 - \tau_2 = -(h_1 + h_2),$$

we have

$$D_2(\tau_0) = -\frac{2}{h_1 + h_2} \left(D_1^{\text{cur}} - D_1^{\text{prev}} \right).$$

Substituting the expressions above gives

$$D_2(\tau_0) = F''(\tau_0) + \frac{2}{h_1 + h_2} \left[-\left(\frac{1}{h_1} + \frac{1}{h_2} \right) R_3^{(1)} + \frac{1}{h_2} R_3^{(2)} \right].$$

Taking the norm and applying the triangle inequality gives

$$\left\| F''(\tau_0) - D_2(\tau_0) \right\|_2 \leq \frac{2}{h_1 + h_2} \Bigg[\left(\frac{1}{h_1} + \frac{1}{h_2} \right) \left\| R_3^{(1)} \right\|_2 + \frac{1}{h_2} \left\| R_3^{(2)} \right\|_2 \Bigg].$$

Applying the remainder bounds then yields

$$\left\| F''(\tau_0) - D_2(\tau_0) \right\|_2 \leq C_{\text{bound}}^{(2)}(h_1, h_2) M_3,$$

where

$$C_{\text{bound}}^{(2)}(h_1, h_2) = \frac{1}{3(h_1 + h_2)} \left[h_1^2 + \frac{h_1^3}{h_2} + \frac{(h_1 + h_2)^3}{h_2} \right].$$

Unlike the fixed-spacing finite-difference case, the coefficient explicitly depends on the actual spacing of the recent computation anchors. In the equal-spacing case $h_1 = h_2 = N$, the bound reduces to

$$C^{(2)}_{\text{bound}}(N, N) = \frac{5}{3}N,$$

showing the expected linear dependence on the anchor spacing for this one-sided, non-centered configuration.

The linear dependence on the anchor spacing reflects the one-sided, non-centered placement of previously observed computation anchors relative to the most recent anchor.

More generally, we use

$$C^{(r)}_{\text{bound}}(h_1, \ldots, h_r)$$

to denote the spacing-dependent coefficient associated with the approximation error of the r-th derivative:

$$\left\| F^{(r)}(\tau_0) - D_r(\tau_0) \right\|_2 \leq C^{(r)}_{\text{bound}}(h_1, \ldots, h_r) M_{r+1}.$$

Since our primary setting uses a maximum prediction order of $m = 2$, we restrict the explicit derivation to the first- and second-order coefficients.

The bounds expose two quantities that govern forecasting reliability: the prediction distance $|k|$ and the computation-anchor configuration. However, they still depend on higher-order derivative magnitudes such as M_2 and M_3, which are not directly available during inference. In the next subsection, we construct observable approximations of these higher-order quantities from cached computation anchors.

A.3 Higher-Order Difference Approximation

The error expressions derived in the previous subsection depend on higher-order derivative magnitudes such as

$$M_{m+1} = \sup_{\xi} \left\| F^{(m+1)}(\xi) \right\|_2,$$

which are not directly available during diffusion inference. To obtain an observable estimate of the local next-order variation, we use one additional full-computation anchor and construct a divided difference one order higher than the Taylor predictor itself.

For an m-th order Taylor predictor, the derivative approximations

$$D_1(\tau_0), \ldots, D_m(\tau_0)$$

are constructed from the most recent $m + 1$ computation anchors

$$\tau_0, \tau_1, \ldots, \tau_m.$$

To estimate the magnitude of the next-order derivative, we additionally retain

$$\tau_{m+1},$$

the preceding full-computation anchor, and compute the $(m + 1)$-th Newton divided difference

$$F[\tau_0, \ldots, \tau_{m+1}].$$

We define the corresponding higher-order derivative approximation as

$$D_{m+1}(\tau_0) = (m + 1)! F[\tau_0, \ldots, \tau_{m+1}].$$

For a sufficiently smooth scalar function, the mean-value property of divided differences implies that there exists some

$$\xi^\star \in [\tau_0, \tau_{m+1}]$$

such that

$$(m + 1)! F[\tau_0, \ldots, \tau_{m+1}] = F^{(m+1)}(\xi^\star).$$

Thus, the $(m + 1)$-th divided difference corresponds to the true $(m + 1)$-th derivative evaluated at some point within the span of the retained computation anchors, rather than being an arbitrary finite-difference quantity.

For the feature tensors encountered in our setting, we compute the divided difference element-wise and use its ℓ_2 norm as a scalar measure of local higher-order variation.

Accordingly, we define

$$\widehat{M}_{m+1} = \|D_{m+1}(\tau_0)\|_2 = (m+1)!\,\|F[\tau_0, \ldots, \tau_{m+1}]\|_2 .$$

Importantly, $\widehat{M}_{m+1}$ is not assumed to be a strict upper bound on

$$M_{m+1} = \sup_{\xi} \left\| F^{(m+1)}(\xi) \right\|_2 .$$

Rather, it serves as an inexpensive observable estimate of the local magnitude of the otherwise unavailable $(m+1)$-th derivative.

For the second-order predictor used in our primary setting, four computation anchors

$$\tau_0, \qquad \tau_1, \qquad \tau_2, \qquad \tau_3$$

are sufficient to construct the third-order derivative estimate. Let

$$D_2^{\mathrm{cur}} = 2!F[\tau_0, \tau_1, \tau_2]$$

and

$$D_2^{\mathrm{prev}} = 2!F[\tau_1, \tau_2, \tau_3].$$

The third-order derivative approximation is then

$$D_3(\tau_0) = \frac{3}{\tau_0 - \tau_3} \left(D_2^{\mathrm{cur}} - D_2^{\mathrm{prev}}\right),$$

which is equivalent to

$$D_3(\tau_0) = 3!F[\tau_0, \tau_1, \tau_2, \tau_3].$$

We therefore use

$$\widehat{M}_3 = \|D_3(\tau_0)\|_2$$

as the observable higher-order variation signal for a second-order Taylor predictor.

Similarly, for a first-order predictor, the second-order quantity

$$D_2(\tau_0) = 2!F[\tau_0, \tau_1, \tau_2]$$

provides the corresponding estimate

$$\widehat{M}_2 = \|D_2(\tau_0)\|_2 .$$

More generally, for derivative-approximation terms of order $r < m$, the required quantity

$$\widehat{M}_{r+1} = \|D_{r+1}(\tau_0)\|_2$$

is already available from the divided differences used by the m-th order predictor and therefore requires no additional computation anchor. For example, when $m = 2$, the quantity $\widehat{M}_2 = \|D_2(\tau_0)\|_2$ is already available from the three anchors used by the second-order predictor, whereas only $\widehat{M}_3$ requires the additional anchor τ_3. Only the highest next-order quantity

$$\widehat{M}_{m+1} = \|D_{m+1}(\tau_0)\|_2$$

requires the additional anchor τ_{m+1}.

The same observable quantities can be combined with different components of the approximation-error analysis to construct several online proxy candidates. We define these variants in the following subsection.

A.4 Error-Derived Proxy Candidates

The approximation-error analysis in section A.2 provides two distinct components: the error introduced by approximating temporal derivatives from irregular computation anchors and the Taylor truncation error. Replacing the unavailable higher-order derivative bounds with the observable divided-difference estimates from section A.3 yields several online risk-proxy candidates.

Taylor truncation proxy. Using only the Taylor truncation component gives

$$p_m^{\mathrm{Taylor}}(t) = \frac{|k|^{m+1}}{(m+1)!}\widehat{M}_{m+1}. \tag{11}$$

This is the proxy used by the final ARCS scheduler in section 3.4.

Derivative-approximation proxy. Retaining the terms associated with approximating the temporal derivatives gives

$$p_m^{\mathrm{DD}}(t) = \sum_{r=1}^{m} \frac{|k|^r}{r!} C_{\mathrm{bound}}^{(r)}(h_1, \ldots, h_r)\widehat{M}_{r+1}. \tag{12}$$

Combined proxy. Using both approximation-error components gives

$$p_m^{\mathrm{Full}}(t) = p_m^{\mathrm{DD}}(t) + p_m^{\mathrm{Taylor}}(t). \tag{13}$$

Because the observable quantities $\widehat{M}_{r+1}$ are local divided-difference estimates rather than guaranteed derivative suprema, none of these quantities is interpreted as a certified upper bound on the true forecasting error. Instead, they are online scheduling signals derived from different components of the same approximation-error analysis.

The final ARCS configuration uses the Taylor truncation proxy in equation 11. The derivative-approximation and combined constructions are retained as alternative proxy variants and evaluated in our ablation study.

Second-order predictor. For the second-order predictor used in our primary setting,

$$m = 2,$$

the Taylor truncation proxy becomes

$$p_2^{\mathrm{Taylor}}(t) = \frac{|k|^3}{6}\widehat{M}_3. \tag{14}$$

The derivative-approximation proxy becomes

$$\begin{aligned} p_2^{\mathrm{DD}}(t) = {} & \frac{|k|h_1}{2}\widehat{M}_2 \\ & + \frac{|k|^2}{6(h_1+h_2)}\left[h_1^2 + \frac{h_1^3}{h_2} + \frac{(h_1+h_2)^3}{h_2}\right]\widehat{M}_3, \end{aligned} \tag{15}$$

and the combined proxy is

$$p_2^{\mathrm{Full}}(t) = p_2^{\mathrm{DD}}(t) + \frac{|k|^3}{6}\widehat{M}_3. \tag{16}$$

The derivative-approximation variants explicitly depend on the recent anchor spacings (h_1, h_2) through the coefficients derived in section A.2. The Taylor truncation proxy used by the final ARCS configuration instead depends on the prediction distance $|k|$ and the observable next-order divided difference $\widehat{M}_3$, which is itself constructed from the irregular computation anchors.

A.5 Scheduler Details

Progressive effective order. During initialization, the available anchor history may be insufficient for the maximum prediction order. ARCS therefore uses the highest effective order for which both the predictor and its corresponding risk proxy can be constructed.

For a zeroth-order predictor,

$$\widehat{F}_0(t) = F(\tau_0),$$

the next-order divided difference is

$$D_1(\tau_0) = F[\tau_0, \tau_1],$$

and we define

$$\widehat{M}_1 = \|D_1(\tau_0)\|_2 .$$

The corresponding Taylor truncation proxy is therefore

$$p_0^{\text{Taylor}}(t) = |k|\widehat{M}_1.$$

For $m = 1$, three computation anchors are required to construct both the first-order predictor and its next-order risk estimate, while for $m = 2$, four computation anchors are required. As additional full-computation anchors become available, ARCS progressively increases the effective order until reaching the configured maximum order.

Cross-order normalization. We estimate a representative risk scale q_m for each effective order m from calibration runs using median statistics. Taking the maximum order $m_{\text{ref}} = m_{\max}$ as the reference, we define

$$a_m = \frac{q_{m_{\text{ref}}}}{q_m}, \qquad a_{m_{\text{ref}}} = 1. \tag{17}$$

The calibration statistics used in this ratio must be positive and finite. The resulting scales are fixed during evaluation.

Module-level risk and adaptive decision. For each forecasted module ℓ, the next-order derivative approximation is reduced to a scalar using its global ℓ_2 norm:

$$\widehat{M}_{m+1,\ell} = \|D_{m+1,\ell}(\tau_0)\|_2 , \tag{18}$$

where the norm is taken over all elements of the derivative approximation tensor, consistent with the magnitude defined in equation 8.

The raw module-level risk is $p_{m,\ell}(t)$ from equation 9. We apply the cross-order scale as

$$\widetilde{p}_{m,\ell}(t) = a_m p_{m,\ell}(t). \tag{19}$$

The raw and scaled timestep-level RMS scores are, respectively,

$$R_m(t) = \sqrt{\frac{1}{L}\sum_{\ell=1}^{L} p_{m,\ell}(t)^2}, \qquad P_m(t) = \sqrt{\frac{1}{L}\sum_{\ell=1}^{L} \widetilde{p}_{m,\ell}(t)^2} = a_m R_m(t). \tag{20}$$

ARCS makes a single shared Full/Forecast decision by comparing $P_m(t)$ with $\gamma(t)$. Computing this risk requires no additional network evaluation.

Table 4: Effect of the threshold-schedule parameter r_γ on DiT-XL/2 at 11 NFC. Values $r_\gamma < 1$, $r_\gamma = 1$, and $r_\gamma > 1$ correspond to decreasing, constant, and increasing thresholds, respectively.

r_γ	FID ↓	sFID ↓	IS ↑
0.90	2.77	5.09	232.37
1.00	2.73	4.96	232.83
1.05	2.71	4.88	**233.27**
1.10	2.71	4.88	233.12
1.15	**2.70**	4.88	233.14
1.20	2.72	**4.86**	232.97

Table 5: Effect of progressive forecasting order and cross-order normalization on DiT-XL/2 at 6 NFC.

Configuration	FID ↓	sFID ↓	IS ↑
ARCS (ours)	**5.17**	**7.65**	**200.09**
w/o order normalization	10.99	11.93	148.02
w/o progressive order	113.01	48.02	10.01

B Additional Ablations

B.1 Threshold Schedule

Table 4 shows that increasing r_γ from 0.90 generally improves generation quality, with similar performance across $r_\gamma = 1.05$–1.20. We use $r_\gamma = 1.15$ as the default setting, which lies within this stable range and achieves the lowest FID while maintaining comparable sFID and IS.

B.2 Progressive Forecasting Order and Proxy Normalization

Table 5 shows that both progressive forecasting order and cross-order normalization contribute to generation quality at the same 6 NFC budget. Removing cross-order normalization substantially degrades all quality metrics, while removing progressive order leads to a much larger degradation. These results support using progressive order together with cross-order normalization when adapting the forecasting order during sampling.

B.3 Module Aggregation

We compare mean, max, and RMS aggregation for combining module-level risk proxies into a shared timestep-level scheduling decision. As shown in Table 6, RMS consistently achieves the best FID, sFID, and IS at the same 11-NFC budget, while max aggregation also performs competitively. These results support RMS as an effective aggregation rule, and we use it as the default in ARCS.

Table 6: Effect of module-level risk aggregation on DiT-XL/2 at 11 NFC.

Aggregation	FID ↓	sFID ↓	IS ↑
Mean	2.80	5.20	232.17
Max	2.71	5.05	232.90
RMS	**2.70**	**4.88**	**233.14**

B.4 Baseline Forecasting-Order Controls

Our main tables use the documented baseline configurations. Because those configurations do not always use the same forecasting order as ARCS, we also perform order-matched controls. For DiT-XL/2, CG-Taylor uses feature forecasting order $m = 1$ by default, while ARCS uses $m = 2$. We

therefore repeat CG-Taylor with $m = 2$, retaining its first-block controller and the same ϵ values. Table 7 reports the result at all three computation budgets.

Table 7: Forecasting-order sensitivity of CG-Taylor on DiT-XL/2. The default feature forecasting order $m = 1$ is compared with an order-matched setting using $m = 2$, consistent with ARCS.

Configuration	Feature order m	ϵ	NFC	FID ↓	sFID ↓	IS ↑
CG-Taylor default	1	0.06	11.0	4.42	4.82	202.55
CG-Taylor order-matched	2	0.06	11.0	3.76	4.82	210.36
CG-Taylor default	1	0.20	8.0	17.68	11.46	124.31
CG-Taylor order-matched	2	0.20	8.0	8.46	6.45	164.92
CG-Taylor default	1	0.60	6.0	73.45	32.18	37.83
CG-Taylor order-matched	2	0.60	6.0	116.41	77.43	10.81

For FLUX.1-dev, SpeCa uses maximum order 2 by default, whereas ARCS uses maximum order 1. We repeat SpeCa with maximum order 1 and choose its base thresholds using realized NFC only. Table 8 reports both the default and order-matched configurations.

Table 8: Forecasting-order sensitivity of SpeCa on FLUX.1-dev. The default maximum order $m = 2$ is compared with an order-matched setting using $m = 1$, consistent with ARCS.

Configuration	Max. order	Base threshold	NFC	PSNR ↑	SSIM ↑	LPIPS ↓
SpeCa default	2	9	10.2	17.62	0.62	0.46
SpeCa order-matched	1	8	10.1	18.00	0.65	0.43
SpeCa default	2	16	7.2	16.10	0.48	0.64
SpeCa order-matched	1	16	7.1	16.03	0.51	0.61

C Implementation and Reproducibility Details

C.1 Experimental Configurations

DiT-XL/2. We use the pretrained DiT-XL/2 checkpoint (Peebles & Xie, 2023) on ImageNet (Deng et al., 2009) at 256×256 resolution, with classifier-free guidance scale 1.5 and DDIM-50 sampling. Final evaluation uses 50,000 images with seed 2. ARCS calibration uses 1,024 samples with seed 0, and a separate 1,024-sample run with seed 1 is used for budget verification. Generation uses batch size 32 on one NVIDIA RTX 5090 GPU.

FLUX.1-dev. We use the official FLUX.1-dev checkpoint (Black Forest Labs, 2024) and 200 DrawBench prompts (Saharia et al., 2022) with 50 sampling steps and guidance scale 3.5. Prompts 0–31 with seed 0 are used for ARCS calibration, and prompts 32–63 with seed 1 for budget verification. Final evaluation uses all 200 prompts with seed 2 at 992×992 resolution and batch size 1 on one NVIDIA RTX 5090 GPU.

HunyuanVideo. We use the official HunyuanVideo FP8 checkpoint (Kong et al., 2024) on two NVIDIA RTX 5090 GPUs with Ulysses sequence-parallel degree 2. We generate 65-frame videos at 480×640 resolution with 50 sampling steps, embedded guidance scale 6.0, and flow shift 7.0. We evaluate 944 unique VBench prompts (Huang et al., 2024). A 32-prompt subset with seed 0 is used for order-scale calibration and a disjoint 32-prompt subset with seed 1 for threshold calibration. Final evaluation uses seeds 2, 3, and 4.

Evaluation metrics. For DiT-XL/2, we report FID (Heusel et al., 2017), sFID (Dhariwal & Nichol, 2021), and IS (Salimans et al., 2016). For FLUX.1-dev, we report ImageReward (Xu et al., 2023), CLIP (Radford et al., 2021), PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). For

HunyuanVideo, we report VBench (Huang et al., 2024), PSNR, SSIM, and LPIPS. For FLUX.1-dev and HunyuanVideo, reconstruction metrics are computed against the Full 50-step outputs generated from the same prompts and seeds.

Baseline configurations. We use the official baseline implementations and select operating points based on realized computation budgets rather than generation quality. For DiT-XL/2, TaylorSeer uses $m = 2$ with fixed intervals $N \in \{5, 7, 10\}$ for the 11-, 8-, and 6-NFC settings. CG-Taylor uses its default feature order $m = 1$, first-block maximum order 3, and $\epsilon \in \{0.06, 0.20, 0.60\}$. SpeCa uses maximum polynomial order 4, with the remaining parameters reported in Table 9.

Table 9: SpeCa configurations for DiT-XL/2. The three operating points were selected using realized computation budgets.

Target	Base threshold	Decay rate	Min. Taylor steps	Max. Taylor steps	Realized NFC
≈ 11	0.2	0.05	3	8	11.0000
≈ 8	0.3	1.00	3	8	8.5905
≈ 6	1.0	1.00	3	11	5.9175

For FLUX.1-dev, TaylorSeer uses $m = 1$ with $N \in \{6, 9\}$. SpeCa uses maximum polynomial order 2, `decay_rate` $= 0.3$, `min_taylor_steps` $= 2$, `max_taylor_steps` $= 10$, and base thresholds 9 and 16. CG-Taylor uses $\epsilon = 0.30$ and 0.80. These settings yield approximately 10- and 7-NFC operating points.

C.2 ARCS Configuration

ARCS uses the Taylor truncation proxy with global absolute-ℓ_2 reduction within each module and RMS aggregation across modules. The maximum forecasting order is $m_{\max} = 2$ for DiT-XL/2 and $m_{\max} = 1$ for FLUX.1-dev and HunyuanVideo. At each timestep, ARCS uses the highest order for which both the predictor and risk proxy are available, falling back to Full computation when the required history is unavailable or the risk estimate is non-finite. No online budget controller is used.

Calibration. Cross-order scales are estimated as described in Appendix A.5. The resulting scales are $(a_0, a_1, a_2) = (0.381574, 0.628927, 1)$ for DiT-XL/2, $(a_0, a_1) = (0.274955, 1)$ for FLUX.1-dev, and $(a_0, a_1) = (0.282423, 1)$ for HunyuanVideo.

We use $r_\gamma = 1.15$ for all models and select γ_0 by matching the target NFC on the calibration set without using generation-quality metrics. For DiT-XL/2, $\gamma_0 \in \{1190, 1825, 2450\}$ targets 11, 8, and 6 NFC, respectively. For FLUX.1-dev, $\gamma_0 \in \{5250, 10500\}$ gives approximately 10- and 7-NFC operating points. All calibrated scales and thresholds remain fixed during evaluation.