Tikhonov-Stabilized Bézier Representation Forecasting for Training-free Diffusion Acceleration

Lei Zhu[1,*] Mujie Lin[1,2,*] Ruochong Zheng[1,2], Guangyi Wang[3]
Hao Li[1], Peng Jin[1], Chang Liu[4], Jie Chen[5,6,1,2,†]

[1]School of Electronic and Computer Engineering, Peking University, Shenzhen, China
[2]AI for Science (AI4S)-Preferred Program, Peking University Shenzhen Graduate School, China
[3]School of Informatics, Xiamen University, China
[4]Department of Automation and BNRist, Tsinghua University, Beijing, China
[5]School of Intelligence Science and Engineering, Harbin Institute of Technology, Shenzhen, China
[6]Pengcheng Laboratory, Shenzhen, China

Abstract

Diffusion models, particularly Diffusion Transformers, achieve strong image and video generation quality but remain expensive at inference time due to repeated denoiser evaluations. Feature caching offers a deployment-friendly acceleration strategy by reusing or predicting intermediate representations without retraining the generator or modifying the sampler. However, existing cache-then-forecast methods often rely on Taylor-style finite-difference extrapolation, which can become unstable over longer cache intervals, and typically predict local module outputs whose errors may accumulate through subsequent denoiser blocks. We propose **BeziCast**, a training-free diffusion acceleration framework that forecasts output-proximal denoising representations using low-order Bézier trajectories. BeziCast estimates Bézier control points via Tikhonov-stabilized fitting, providing a smooth, capacity-controlled temporal parameterization that avoids explicit high-order derivative estimation and decouples trajectory capacity from cache-query density. To moderate aggressive extrapolation, we further derive equivalent forecasting weights and introduce a convex-hull-inspired guardrail that detects high-risk predictions and softly pulls them toward a conservative simplex-projected estimate. Evaluations across image and video diffusion models demonstrate the effectiveness of BeziCast. In particular, BeziCast accelerates FLUX.1 by up to $4.79\times$ and HunyuanVideo by up to $4.11\times$, achieving substantially better reference fidelity while maintaining comparable perceptual and semantic quality relative to competing acceleration baselines.

1 Introduction

Diffusion models[12], especially large generative models based on Diffusion Transformer[32, 4, 56, 15], have become a standard approach for high-fidelity image and video generation. Their sampling process, however, still requires repeated denoiser evaluations across many timesteps, and this inference cost remains a major obstacle to large-scale deployment.

A central question in diffusion acceleration is how to approximate the original multi-step sampling process under a limited NFE budget[25, 50]. Existing approaches can be broadly grouped into three directions. Solver-based methods improve the sampler through higher-order updates[25, 26, 50], timestep schedules[14], and trajectory-aware error correction or time allocation[54, 41, 3, 55, 57, 40,

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

53, 30]. Distillation-based methods compress multi-step sampling into few-step or one-step generation through progressive distillation[34], consistency modeling[27], or distribution matching[47]. Feature caching methods instead keep the pretrained generator largely unchanged and reduce the cost of each denoiser evaluation by reusing or predicting internal representations[28, 21]. Compared with the former two directions, feature caching is particularly deployment-friendly because it avoids retraining the full model and does not require redesigning the sampling dynamics.

Early training-free feature caching methods mainly followed a *cache-then-reuse* paradigm, where intermediate features from nearby timesteps are directly reused to skip selected layers or modules[28, 35, 24, 29]. Recent methods move toward *cache-then-forecast*. TaylorSeer[21] predicts future features with finite differences and Taylor expansion, and SpeCa[22] adds a lightweight verifier to accept or reject predicted features. Spectrum[10] takes a global function-approximation view by fitting Chebyshev polynomial coefficients for final-layer denoiser features and deriving a horizon-independent error bound under its approximation setting. These methods show that denoiser representations have exploitable temporal regularity, but they also leave two design choices: which representation should be predicted, and which temporal basis should be used.

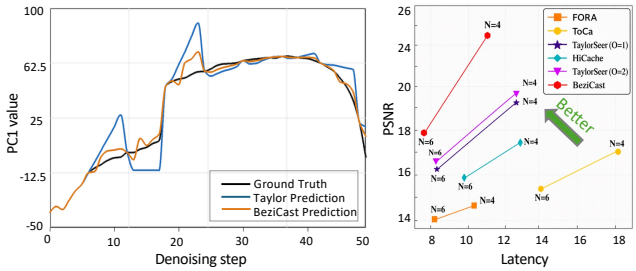


Figure 1: **Stable, Fast, and Accurate Diffusion Acceleration.** *Left*: PC1 projections of output-proximal representation in denoising step. *Right*: Speedup and PSNR comparison, where $\mathcal{N}$ is the cache interval. BeziCast achieves faster inference while better quality.

The first design choice is the **temporal basis**. Taylor-style predictors are simple, but finite-difference derivative estimates and local power-basis extrapolation can amplify small cache perturbations at longer offsets or higher orders[21, 17, 7], as evidenced by the instability of the blue curve and its large deviation from the ground truth on the left of Fig. 1. This sensitivity is consistent with the numerical behavior of monomial and Vandermonde-type systems[36], the non-monotone feature trajectories discussed by Feng et al.[7], and the large-horizon error growth observed by Han et al.[10]. At the same time, diffusion sampling trajectories often exhibit structured low-dimensional geometry[2]. We therefore seek a compact temporal parameterization that avoids explicit high-order finite differences while remaining analytically evaluable at cache steps.

The second design choice is the **forecasting target**. Existing caching methods often predict attention outputs, MLP outputs inside transformer blocks[35, 29, 48, 22]. These local targets give fine-grained control, but their errors still pass through downstream denoiser blocks before reaching the final noise estimate and may accumulate across predicted modules. We instead predict an output-proximal denoising representation, which makes the path from representation error to denoising error shorter and more direct without assuming that this representation is intrinsically easier to forecast.

Based on these observations, we propose **BeziCast**, **a training-free *Tikhonov-stabilized Bézier representation forecasting* framework, achieving faster, more stable, and more accurate acceleration than competing methods**, as shown on the right of Fig. 1. BeziCast predicts an output-proximal denoising representation with a low-order Bernstein/Bézier trajectory whose control points are estimated by Tikhonov-stabilized fitting. The Bézier form provides analytic evaluation and a compact capacity-controlled function family[6, 42, 31], while the Bernstein basis has classical stability properties on a fixed interval[5], as shown by the orange curve on the left of Fig. 1, which is closer to the ground truth than Taylor-based extrapolation. Because cache queries are often short-range extrapolations, BeziCast does not use the convex-hull property as a guarantee; instead, it uses equivalent weights to detect signed or non-affine extrapolation and pulls high-risk predictions toward a simplex-projected conservative estimate. The pretrained denoiser and sampler remain unchanged.

Our contributions are as follows: (1) We formulate training-free cache-then-forecast acceleration as output-proximal representation forecasting, avoiding multiple local module targets. (2) We use a low-order Bernstein/Bézier trajectory with Tikhonov-stabilized control-point fitting, replacing finite-difference Taylor extrapolation, decoupling trajectory capacity from cache-query density, and expressing each prediction as a regularized linear combination of cached representations. (3) We

introduce an equivalent-weight convex-hull guardrail that detects aggressive extrapolation through negative weight mass, deviation from unit sum, and the ℓ_1 norm, then pulls high-risk predictions toward a simplex-projected conservative estimate.

2 Related Work

Diffusion acceleration methods mainly reduce either the number of denoising steps, the cost of each denoising step, or both.

Sampler-based Acceleration. Sampler-based methods reduce the number of function evaluations by improving the reverse diffusion discretization. DDIM [37] enables deterministic non-Markovian sampling from DDPM [12] models with fewer steps. DPM-Solver [25] and DPM-Solver++ [26] derive high-order ODE solvers for probability-flow sampling, UniPC [50] unifies predictor-corrector updates, and EDM [14] improves noise parameterization, preconditioning, and sampling schedules. Recent trajectory-aware methods further exploit path regularity, error correction, or time allocation, including AMED-Solver [54], Trajectory Regularity [3], Simple and Fast Distillation [55], PAS [41], PFDiff [40], fast-sampling analysis [53], and BézierFlow [30]. These methods mainly accelerate sampling by changing the solver, schedule, or trajectory correction strategy.

Training-based Acceleration. Training-based methods compress a multi-step teacher into a faster generator through extra optimization. Progressive Distillation [34] halves sampling steps iteratively; Consistency Models [38] and Latent Consistency Models [27] enforce trajectory-consistent predictions for few-step generation; DMD [47] and DMD2 [46] match the teacher distribution rather than the full teacher trajectory; and InstaFlow [23] uses rectified-flow-style training to straighten text-to-image generation paths. These approaches can be effective, but they usually require additional training, teacher models, or specialized objectives.

Feature-caching-based Acceleration. Feature-caching methods[21, 43, 35, 44] reduce inference cost by exploiting temporal redundancy in denoiser activations while keeping the pretrained denoiser largely unchanged. Early cache-then-reuse methods directly reuse intermediate features or residuals: DeepCache [28] reuses high-level U-Net features, FORA [35] extends fast-forward caching to diffusion transformers, ToCa [58] introduces token-wise feature caching, and Reuse and Diffuse [8] plus MagCache [29] study feature or residual reuse for video diffusion. Recent cache-then-forecast methods model feature evolution instead of copying old features: TaylorSeer [21] uses finite differences and Taylor expansion, FreqCa [20] adds frequency-aware caching, Forecast-then-Calibrate [51] introduces feature calibration, HiCache [7] replaces Taylor-style monomial forecasting with Hermite-polynomial-based prediction, and Spectrum [10] fits global spectral approximations. Adaptive variants such as TeaCache [19], DiCache [1], SpeCa [22], and token-level cache updates [52] further show that diffusion features exhibit strong temporal and structural redundancy. BeziCast follows the training-free cache-then-forecast line, but focuses on output-proximal representation forecasting with a Tikhonov-stabilized Bézier trajectory and an equivalent-weight guardrail for extrapolation risk.

3 Method

BeziCast is a training-free cache-then-forecast method for diffusion sampling. We recast feature prediction as output-proximal representation trajectory approximation: from a short history of denoising representations near the output, the method estimates a low-order Bézier trajectory and evaluates it at skipped cache steps. As shown in Fig. 2, BeziCast runs the original denoiser at selected full steps, caches output-proximal representations, predicts them at cache steps with Tikhonov-stabilized Bernstein fitting, and applies the guardrail when needed. It does not train additional models, modify the sampler, or change the pretrained denoiser. This section first defines the cache-then-forecast setting, then gives the predictor and guardrail. The full inference algorithm is provided in Appendix A.4.

3.1 Cache-Then-Forecast Formulation

Definition 1 (diffusion sampling). Let $\mathcal{T} = (t_0, t_1, \ldots, t_{T-1})$ be the grid of T sampling steps, where $s \in \{0, \ldots, T-1\}$ is the solver-step index. Given latent state x_s, condition c, and diffusion time t_s, the pretrained denoiser predicts $\epsilon_s = \epsilon_\theta(x_s, t_s, c)$, with fixed parameters θ. Here ϵ_s denotes the sampler-required output, such as noise or velocity. The sampler updates the latent by

$$x_{s+1} = \Psi_s(x_s, \epsilon_s), \tag{1}$$

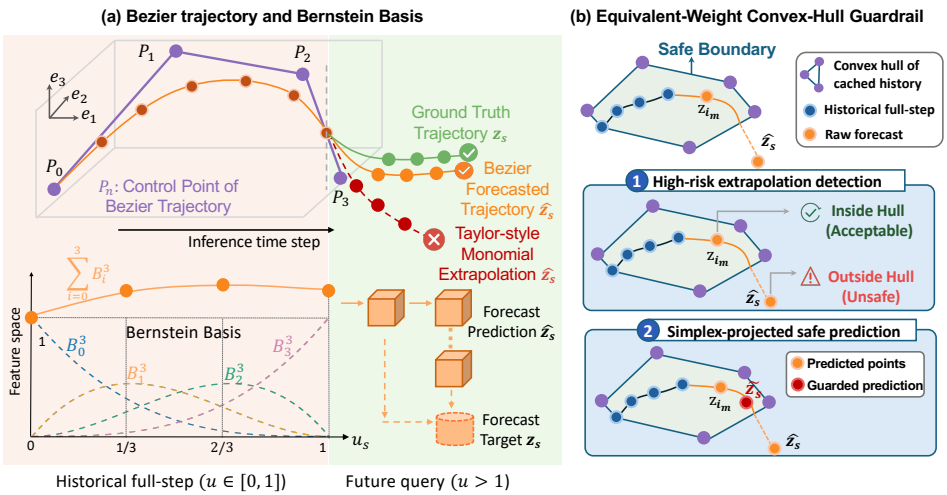


Figure 2: **Overview of BeziCast.** (a) A local cubic Bernstein/Bézier trajectory is fitted from cached full-step representations and evaluated at normalized cache-query coordinates u_s, typically beyond the fitting interval, providing a low-order alternative to Taylor-style extrapolation. (b) BeziCast detects risky extrapolation via equivalent history weights, constructs a simplex-projected prediction inside the cached-representation convex hull, and softly blends it with the raw forecast to obtain the guarded prediction.

where Ψ_s denotes one solver step. We seek to reduce expensive denoiser computation while keeping both the pretrained denoiser ϵ_θ and the sampler update Ψ_s unchanged.

A key design choice in cache-then-forecast acceleration is the representation to cache and predict. Prior works typically forecast local features such as attention, MLP, or residual outputs within transformer blocks [35, 29, 49, 22], but errors in these intermediate targets are propagated through subsequent nonlinear layers and can accumulate if multiple predictions occur per denoising step. To ensure errors are directly reflected in the final noise estimate and to simplify control, we forecast a single output-proximal representation immediately before the final output map, aligning with the final-block forecasting setting of Spectrum [10].

Definition 2 (output-proximal representation). We decompose the pretrained denoiser into a backbone and a final output map:

$$z_s = \mathcal{F}_\theta(x_s, t_s, c), \qquad \epsilon_\theta(x_s, t_s, c) = \mathcal{O}_\theta(z_s, t_s, c). \tag{2}$$

Here $z_s \in \mathbb{R}^{1\times D}$ is a flattened row vector, and D absorbs token and channel dimensions. All representation vectors use this row convention; unsubscripted vector norms are Euclidean. The map $\mathcal{F}_\theta$ denotes the expensive backbone computation up to this representation, while $\mathcal{O}_\theta$ denotes the lightweight output map that converts z_s into the denoising prediction. The output map is always executed, both at full steps and cache steps.

Proposition 1 (output-proximal error path). Assume $\mathcal{O}_\theta$ is Lipschitz on a neighborhood containing both the exact and predicted representations. If $\hat{z}_s = z_s + e_s^z$, with prediction error e_s^z, is an output-proximal prediction, then

$$\|\mathcal{O}_\theta(\hat{z}_s, t_s, c) - \mathcal{O}_\theta(z_s, t_s, c)\| \le L_{\mathcal{O}} \|e_s^z\|, \tag{3}$$

where $L_{\mathcal{O}}$ is a local Lipschitz constant of the output map.

Proposition 1 relates forecasting error directly to output-map sensitivity. Appendix A.2 gives the proof and the corresponding intermediate-layer and sampler-level bounds.

Definition 3 (training-free cache forecaster). Let

$$\mathcal{S}_{\text{full}} \subseteq \{0, \dots, T-1\}, \qquad \mathcal{S}_{\text{cache}} = \{0, \dots, T-1\} \setminus \mathcal{S}_{\text{full}} \tag{4}$$

denote full-step and cache-step indices. At a full step $s \in \mathcal{S}_{\text{full}}$, the method runs the original denoiser backbone, obtains z_s and ϵ_s by Eq. (2), and stores z_s. At a cache step $s \in \mathcal{S}_{\text{cache}}$, it skips the backbone, predicts a raw representation $\hat{z}_s$ from cached representations, optionally forms the guarded

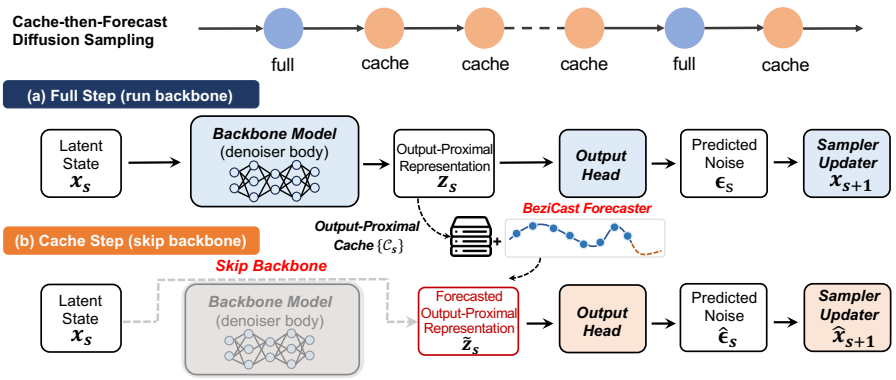


Figure 3: BeziCast sampling pipeline. Full steps run the backbone and cache output-proximal representations. Cache steps use Tikhonov-stabilized Bézier fitting and the guardrail to obtain the representation $\tilde{z}_s$, then run the unchanged output head and sampler update.

representation $\tilde{z}_s$, and computes $\hat{\epsilon}_s = \mathcal{O}_\theta(\tilde{z}_s, t_s, c)$ with the unchanged output map. Without guardrail activation, $\tilde{z}_s = \hat{z}_s$. The sampler then uses $\hat{\epsilon}_s$ in Eq. (1).

Let $\mathcal{C}_s = \{(\tau_{i_m}, z_{i_m})\}_{m=1}^{K}$ be the local cache at step s, where $i_1 < \cdots < i_K < s$ are the most recent $K \le K_{\max}$ full steps, with $K_{\max}$ the history cap. The time coordinate τ_i increases strictly with i, e.g., $\tau_i = -t_i$ for decreasing diffusion times. The available length K may vary with s, and we require $K \ge 2$ for a nondegenerate local time window. A cache forecaster has the form

$$\hat{z}_s = f(\mathcal{C}_s, \tau_s), \qquad s \in \mathcal{S}_{\text{cache}}. \tag{5}$$

Different cache-then-forecast methods correspond to different choices of f. Direct reuse sets $f(\mathcal{C}_s, \tau_s) = z_{i_K}$. Taylor-style forecasting estimates finite differences from nearby cached representations and extrapolates them to τ_s. BeziCast instead uses the regularized Bernstein/Bézier forecaster defined below.

In our cubic Bézier setting, we use a warm-up plus fixed-period schedule (see Appendix A.3). The first W solver steps initialize the history (normally $W \ge 4$, with shorter warm-ups evaluated in the appendix). Thereafter, one full evaluation is performed every $R \ge 1$ steps, where R is the fixed full-step period, and each intermediate cache step is forecast from at most $K_{\max} = 6$ recent full-step representations.

3.2 Tikhonov-Stabilized Bézier Forecaster

The temporal basis determines how cached denoising representations are predicted at skipped timesteps. Taylor-style predictors estimate finite differences and extrapolate a local polynomial. BeziCast uses a low-order Bernstein/Bézier trajectory with Tikhonov-stabilized fitting. The forecaster is specified below by local time normalization, a Bernstein basis, and a Tikhonov control-point estimate. Appendix A.1.1 gives the design motivation and perturbation discussion.

Since future full-step representations are unavailable during online inference, cache-step prediction is typically performed by extrapolating beyond the recent history window. The standard Bézier convex-hull property only holds inside the interpolation interval, so we control extrapolation by clipping the query coordinate with a fixed cap $u_{\max} \ge 1$ and by applying the equivalent-weight guardrail in Section 3.3.

Definition 4 (local time normalization). For a cache step s, let $\mathcal{I}_s = \{i_1, \ldots, i_K\}$, with $i_1 < \cdots < i_K < s$, be the indices of the most recent K full steps. Historical coordinates are normalized by

$$u_m^{\text{hist}} = \frac{\tau_{i_m} - \tau_{i_1}}{\tau_{i_K} - \tau_{i_1} + \varepsilon_\tau}, \qquad m = 1, \ldots, K, \tag{6}$$

where $\varepsilon_\tau > 0$ is a small numerical constant. The cache-step query coordinate is normalized with the same local window and clipped by $u_{\max}$:

$$\bar{u}_s = \frac{\tau_s - \tau_{i_1}}{\tau_{i_K} - \tau_{i_1} + \varepsilon_\tau}, \qquad u_s = \min(\bar{u}_s, u_{\max}), \tag{7}$$

Here $u_{\max} \geq 1$ caps extrapolation; $\bar{u}_s$ is the unclipped query.

Definition 5 (Bernstein/Bézier trajectory). For n control representations $P_0, \ldots, P_{n-1} \in \mathbb{R}^{1\times D}$, define a trajectory of degree at most $n-1$:

$$\hat{z}(u) = \sum_{j=0}^{n-1} b_{j,n-1}(u) P_j, \qquad b_{j,n-1}(u) = \binom{n-1}{j} u^j (1-u)^{n-1-j}. \tag{8}$$

The control representations parameterize a local trajectory; they are fitted parameters and need not coincide with any cached representation.

Definition 6 (Tikhonov control-point fitting). Let $b(u) = [b_{0,n-1}(u), \ldots, b_{n-1,n-1}(u)]^\top \in \mathbb{R}^n$, and construct the Bernstein design matrix $A \in \mathbb{R}^{K\times n}$ by $A_{m,j+1} = b_{j,n-1}(u_m^{\text{hist}})$, for $m = 1, \ldots, K$ and $j = 0, \ldots, n-1$. Stack the cached output-proximal representations as

$$Z = \begin{bmatrix} z_{i_1}^\top & \cdots & z_{i_K}^\top \end{bmatrix}^\top \in \mathbb{R}^{K\times D}. \tag{9}$$

We estimate $P \in \mathbb{R}^{n\times D}$, whose row $j+1$ is P_j, by

$$P_\lambda^\star = \arg\min_P \|AP - Z\|_F^2 + \lambda \|\Gamma P\|_F^2, \tag{10}$$

where $\lambda > 0$ is the regularization strength and Γ is the Tikhonov matrix with n columns and a compatible number of rows; $\|\cdot\|_F$ is the Frobenius norm. We set $\Gamma = I_n$, the $n \times n$ identity (written I below), yielding the ridge-stabilized solution

$$P_\lambda^\star = (A^\top A + \lambda I)^{-1} A^\top Z. \tag{11}$$

Proposition 2 (equivalent linear smoother). With $\Gamma = I$, the cache-step prediction can be written as

$$\hat{z}_s = b(u_s)^\top (A^\top A + \lambda I)^{-1} A^\top Z. \tag{12}$$

Equivalently,

$$\hat{z}_s = \omega_\lambda(u_s)^\top Z, \qquad \omega_\lambda(u_s)^\top = b(u_s)^\top (A^\top A + \lambda I)^{-1} A^\top \in \mathbb{R}^{1\times K}. \tag{13}$$

Proof. Substituting Eq. (11) into Eq. (8) gives Eq. (12). Collecting the coefficients applied to Z gives Eq. (13). Eq. (13) rewrites the prediction as a weighted sum of cached representations. The equivalent weights depend on the local coordinates, the Bernstein basis, and λ. The local-step dependence of A, Z, and ω_λ is implicit.

Proposition 3 (regularization-controlled sensitivity). With the history coordinates, query, and λ fixed, a cache perturbation δZ induces a raw-prediction change $\delta\hat{z}_s$ satisfying

$$\|\delta\hat{z}_s\|_2 \leq \frac{\|b(u_s)\|_2}{2\sqrt{\lambda}} \|\delta Z\|_F. \tag{14}$$

On the clipped query range $u_s \in [0, u_{\max}]$, the factor $\|b(u_s)\|_2$ is bounded by $\sup_{u\in[0,u_{\max}]} \|b(u)\|_2$. The proof and the comparison with ordinary least squares are given in Appendix A.1.

3.3 Equivalent-Weight Convex-Hull Guardrail

The equivalent weights in Eq. (13) are also used to define a coefficient-space score. When $u_s > 1$, the induced historical weights may become signed, deviate from unit sum, or have a large ℓ_1 norm. Tikhonov fitting controls numerical sensitivity, while the guardrail controls the geometry of the historical combination. Indeed, increasing λ drives the equivalent weights toward zero rather than a unit-sum combination (Proposition A.10). The following definitions specify the score and the simplex-projected fallback.

Definition 7 (equivalent-weight risk). Let $w = \omega_\lambda(u_s) \in \mathbb{R}^K$, and write $\mathbf{1} \in \mathbb{R}^K$ for the all-ones vector. The local risk score at cache step s is

$$r_s = |\mathbf{1}^\top w - 1| + \big(\|w\|_1 - |\mathbf{1}^\top w|\big). \tag{15}$$

The first term measures deviation from an affine combination. The second term measures excess signed mass.

Proposition 4 (risk-score interpretation). The score r_s is nonnegative. Moreover, $r_s = 0$ if and only if w lies in the probability simplex

$$\Delta = \{\alpha \in \mathbb{R}^K : \alpha_m \geq 0,\ \mathbf{1}^\top \alpha = 1\}. \tag{16}$$

If $\mathbf{1}^\top w = 1$, then

$$r_s = \|w\|_1 - 1 = 2 \sum_{m=1}^{K} \max(-w_m, 0). \tag{17}$$

The proof is given in Appendix A.6.1.

Definition 8 (guarded prediction). We project the equivalent weights onto the simplex:

$$w_{\text{safe}}(u_s) = \Pi_\Delta(w), \tag{18}$$

where Π_Δ denotes Euclidean projection. The conservative representation is

$$z_s^{\text{safe}} = w_{\text{safe}}(u_s)^\top Z. \tag{19}$$

The final cache-step representation is

$$\tilde{z}_s = (1 - \beta_s)\hat{z}_s + \beta_s z_s^{\text{safe}}, \tag{20}$$

with

$$\beta_s = \beta_{\max} \cdot \text{clip}\left(\frac{r_s - \rho}{\gamma}, 0, 1\right). \tag{21}$$

Here $\rho \geq 0$ is the activation threshold, $\gamma > 0$ is the transition width, $\beta_{\max} \in [0, 1]$ is the maximum strength, and $\text{clip}(a, 0, 1) = \min(1, \max(0, a))$.

Proposition 5 (convex-hull fallback). The representation z_s^{safe} lies in the convex hull of the cached representations $\{z_{i_m}\}_{m=1}^K$. If $\beta_s = 1$, the final prediction $\tilde{z}_s$ equals this convex-hull fallback. If $0 < \beta_s < 1$, the final prediction is pulled toward the fallback but is not forced to lie inside the historical convex hull.

The proof is given in Appendix A.6.1.

4 Experiments

We conduct experiments across image and video diffusion settings, using FLUX.1-dev[4] for text-to-image generation and HunyuanVideo[15] for text-to-video generation as the main benchmarks. We further evaluate matched-NFE comparisons, controlled basis substitutions, guardrail behavior, additional video backbones, and a DDIM-based class-conditional image setting. Each comparison uses the same prompts, initial noise, sampling grid, and model checkpoint within a setting, enabling the results to separate forecasting quality from changes in the denoising budget.

4.1 Experimental Setup

Models and Tasks. For text-to-image generation, we evaluate BeziCast on FLUX.1-dev [4] using a 50-step reference schedule and the 200-prompt DrawBench suite [33], generating images at 1024×1024. For text-to-video generation, we use HunyuanVideo [15] and follow the official VBench [13] prompt set and evaluation protocol. For matched comparisons, all methods share the same prompt set, initial noise, checkpoint, sampler, and time grid. We use five paired random seeds, generating five samples per prompt, and report seed-level uncertainty in Appendix A.7.

Baselines and Metrics. We compare BeziCast with reduced-step sampling, cache-reuse methods FORA [35], ToCa [58], and TeaCache [19], and forecasting methods TaylorSeer [21], HiCache [7], and Spectrum [10]. For BeziCast, $\mathcal{N} = R$ is the full-step period, n the control-point count, and λ the regularization strength. NFE counts full denoiser evaluations, including warm-up but excluding cache steps that run only the output head. We report sampling latency and speedup relative to the 50-step sampler, together with PSNR, SSIM, LPIPS, ImageReward, and CLIP Score [11] for images, and PSNR, SSIM, LPIPS, and VBench Quality Score for videos. PSNR, SSIM, and LPIPS measure reference fidelity to the corresponding 50-step output, whereas ImageReward, CLIP, and VBench evaluate preference, text alignment, or overall perceptual quality. BeziCast's largest quality gains are in reference fidelity; differences in ImageReward and VBench are more modest. These complementary metric groups distinguish preservation of the original output from overall sample quality.

Table 1: Image/video benchmarks against the 50-step reference (†). Parameters: cache interval $\mathcal{N}$, forecast order O, ToCa caching ratio $\mathcal{R}$, TeaCache threshold δ, HiCache Hermite scale σ, and Spectrum schedule growth α. Slash-separated settings give image/video values.

Method	FLUX.1 [dev] / DrawBench							HunyuanVideo / VBench					
	Acceleration		Quality			Image Reward↑	CLIP↑	Acceleration		Quality			VBench Quality↑
	Latency(s)↓	Speedup↑	PSNR↑	SSIM↑	LPIPS↓			Latency(s)↓	Speedup↑	PSNR↑	SSIM↑	LPIPS↓	
50 steps†	37.681	1.00×	–	–	–	1.052	27.01	489.18	1.00×	–	–	–	84.03
25 steps	19.583	1.92×	17.350	0.679	0.228	0.998	27.23	252.40	1.94×	21.779	0.885	0.243	83.60
15 steps	12.379	3.04×	15.191	0.602	0.329	0.953	27.38	157.06	3.12×	18.894	0.796	0.382	83.47
FORA ($\mathcal{N}=4$)	11.208	3.36×	14.702	0.584	0.356	0.885	27.20	153.67	3.18×	17.892	0.740	0.421	83.21
ToCa ($\mathcal{N}=4$, $\mathcal{R}=0.9$)	17.995	2.09×	16.560	0.599	0.251	0.974	27.01	336.76	1.46×	17.837	0.736	0.425	83.21
TeaCache ($\delta=0.4/0.2$)	20.349	1.86×	17.100	0.632	0.228	1.002	**27.32**	178.64	2.74×	19.869	0.796	0.371	83.46
TaylorSeer ($\mathcal{N}=4$, $O=1$)	12.362	3.05×	19.158	0.720	0.149	**1.041**	27.00	152.68	3.20×	20.629	0.877	0.281	82.81
TaylorSeer ($\mathcal{N}=4$, $O=2$)	12.531	3.00×	19.486	0.725	0.144	1.035	26.99	152.87	3.12×	20.475	0.869	0.290	82.25
HiCache ($\mathcal{N}=4$, $O=2$, $\sigma=0.5$)	12.654	2.99×	17.511	0.666	0.206	1.010	26.89	149.63	3.27×	20.866	0.874	0.274	83.06
Spectrum ($\alpha=0.75$)	14.698	2.56×	21.785	0.743	0.097	1.032	26.87	147.32	3.32×	23.182	0.891	0.246	83.31
BeziCast ($\mathcal{N}=4/5$, $n=4$, $\lambda=0.5$)	**10.855**	**3.47×**	**22.382**	**0.782**	**0.090**	1.035	27.17	**135.82**	**3.60×**	**24.783**	**0.916**	**0.197**	**83.48**
FORA ($\mathcal{N}=6$)	8.321	4.52×	14.056	0.551	0.408	0.798	26.96	116.10	4.21×	16.883	0.652	0.504	82.55
ToCa ($\mathcal{N}=6$, $\mathcal{R}=0.9$)	14.342	2.64×	15.037	0.579	0.276	0.965	27.07	230.32	2.13×	16.861	0.648	0.504	82.39
TeaCache ($\delta=0.8/0.25$)	12.833	2.94×	15.106	0.621	0.289	0.960	27.03	157.27	3.11×	18.768	0.721	0.423	83.01
TaylorSeer ($\mathcal{N}=6$, $O=1$)	8.793	4.31×	15.617	0.621	0.286	0.940	26.41	125.36	3.90×	17.561	0.784	0.419	82.80
TaylorSeer ($\mathcal{N}=6$, $O=2$)	8.862	4.27×	15.872	0.598	0.281	**0.988**	26.52	125.54	3.90×	18.041	0.800	0.400	82.52
HiCache ($\mathcal{N}=6$, $O=2$, $\sigma=0.5$)	9.663	3.90×	15.618	0.585	0.296	0.971	26.54	123.47	3.96×	19.143	0.827	0.361	82.29
Spectrum ($\alpha=3.0$)	9.542	3.97×	16.385	0.601	0.267	0.978	26.49	124.32	3.94×	21.236	0.851	0.298	82.17
BeziCast ($\mathcal{N}=6$, $n=4$, $\lambda=0.5$)	**7.881**	**4.79×**	**17.204**	**0.625**	**0.240**	0.987	**27.09**	**118.62**	**4.11×**	**23.631**	**0.904**	**0.217**	**83.21**

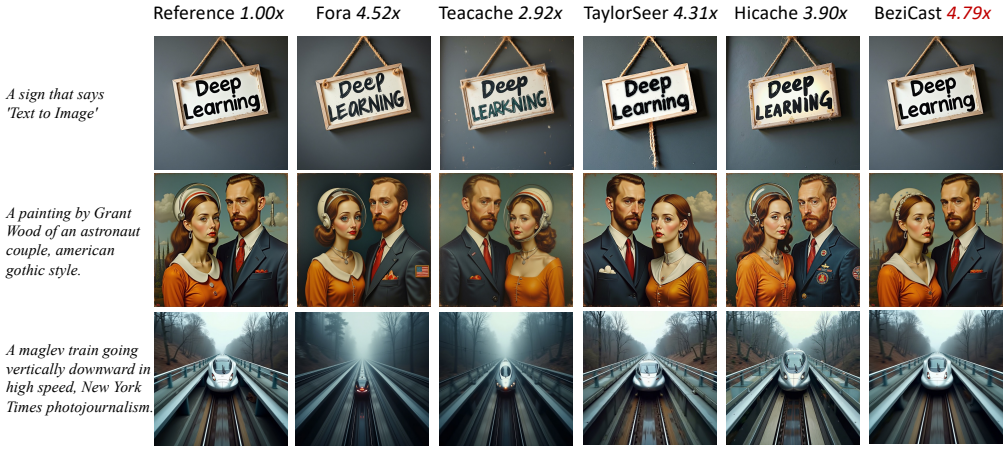


Figure 4: Qualitative comparison on text-to-image generation using FLUX.1. BeziCast delivers the highest speedup while preserving prompt alignment, text readability, and visual fidelity.

4.2 Main Comparisons.

Text-to-Image Generation. The left part of Table 1 reports the quantitative results on FLUX.1-dev. BeziCast achieves a strong speed–fidelity trade-off under both caching intervals. With $\mathcal{N}=4$ and $\lambda=0.5$, it reaches a $3.47\times$ speedup and the best reference fidelity among accelerated methods. Under the more aggressive $\mathcal{N}=6$ setting, it reaches $4.79\times$ speedup with better PSNR, SSIM, and LPIPS than the corresponding reuse and forecasting baselines. These main benchmark configurations are distinct from the identical-schedule comparison below. Fig. 4 provides a qualitative comparison. BeziCast achieves the highest speedup while preserving prompt alignment, text legibility, and global visual fidelity comparable to the full FLUX.1 baseline.

Text-to-Video Generation. On HunyuanVideo (Table 1, right), BeziCast with $n=4, \lambda=0.5$ achieves $3.60\times$ speedup at $\mathcal{N}=5$ and $4.11\times$ at $\mathcal{N}=6$, preserving higher reference fidelity than TaylorSeer and HiCache at comparable speedups. Fig. 5 shows closer agreement with the 50-step reference and fewer visible artifacts than TaylorSeer.

4.3 Matched-Budget and Mechanistic Validation

Matched-NFE comparison. To isolate forecasting overhead from the denoising budget, we fix the 50-step grid, $\mathcal{N}=6$, $W=5$, and $\mathcal{S}_{\text{full}}=\{0,1,2,3,4,10,16,22,28,34,40,46\}$ for TaylorSeer,

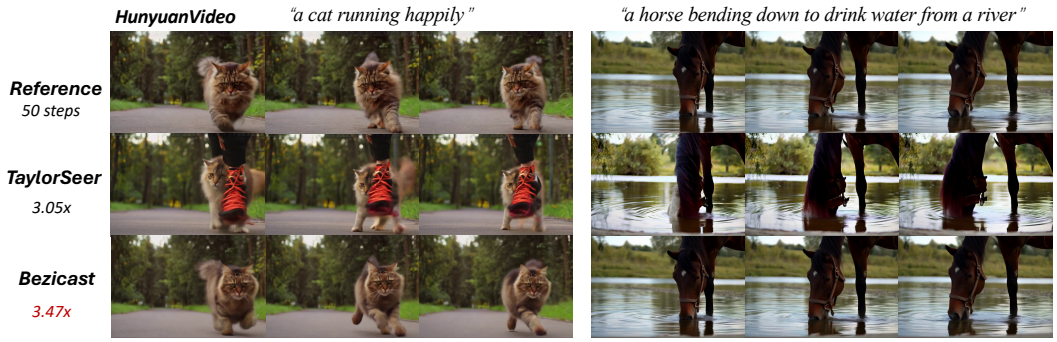


Figure 5: Qualitative comparison on text-to-video generation using HunyuanVideo. BeziCast closely follow the 50-step reference, whereas TaylorSeer is slower and introduces visible artifacts.

Table 2: Matched-NFE comparison with identical full-step locations and 12 full denoiser evaluations.

Model	Method	PSNR ↑	SSIM ↑	LPIPS ↓	Latency ↓
FLUX.1-dev	TaylorSeer	15.987	0.610	0.272	8.982
	HiCache	15.727	0.596	0.283	9.742
	BeziCast	**17.022**	**0.617**	**0.251**	**7.724**
HunyuanVideo	TaylorSeer	18.152	0.809	0.387	128.76
	HiCache	19.213	0.831	0.347	127.01
	BeziCast	**23.221**	**0.883**	**0.236**	**116.04**

HiCache, and BeziCast. All three therefore use exactly 12 full evaluations. Table 2 shows improved reference fidelity and lower latency for BeziCast on both models. Appendix A.7 gives the timing protocol and multi-seed results.

Mechanistic evidence for the guardrail. In the diagnostic schedules, the risk-adaptive guardrail activates at 31.25%–45.71% of cache positions on FLUX and 35.29%–42.86% on HunyuanVideo, with mean activated blending strengths below 0.5. Its risk score correlates positively with position-averaged representation and denoising-output errors (Spearman correlation 0.570–0.862), and activated positions have 1.63–2.51 times larger output error than inactive positions. The paired ablation in Appendix A.9 reduces HunyuanVideo worst-10% LPIPS from 0.362 to 0.250 while changing latency from 44.64s to 44.75s.

Basis and global-versus-local comparisons. Under a fixed local history, degree, regularization coefficient, and 15-NFE schedule, local Bernstein forecasting reduces output error from 0.247 to 0.211 and the realized equivalent-weight norm from 2.12/3.20 to 1.60/2.87 (mean/max), while improving PSNR, SSIM, and LPIPS at nearly identical latency. In a separate 14-NFE end-to-end comparison, BeziCast reaches 11.58s versus 14.30s for Spectrum and improves PSNR from 22.77 to 23.38. These results distinguish the local fixed-history forecasting design from a global spectral approximation; the controlled comparisons are reported in Appendix A.8.

Additional architectures and samplers. The same trend extends to CogVideoX1.5-5B, LTX-Video-2B, and Wan2.1-T2V-14B, where BeziCast achieves the best speedup and reference fidelity among the compared methods. These auxiliary rows use common subsets of the corresponding evaluation data, with identical samples and random seeds within each model; their video lengths are 65, 161, and 81 frames, respectively. On DiT-XL/2 with a 50-step DDIM sampler, BeziCast reaches FID 2.53 and sFID 4.84 at 12 NFE. Appendix A.10 reports these results and the comparison with 20- and 10-step DPM-Solver++.

4.4 Ablation Studies

Projection and blending. Table 3 separates the conservative simplex prediction from the blending policy. We compare the raw forecast, hard projection with $\beta_s = 1$ at activated positions, a fixed soft blend using the full method's mean activated strength, and risk-adaptive blending. The fixed

Table 3: Projection and blending ablation with paired prompts, noise, schedules, and forecasting parameters. Hard and fixed blending use the same activation rule as the full method. W10 denotes mean LPIPS over the worst 10% of samples; P95 denotes the 95th-percentile denoising-output forecasting error.

Variant	PSNR ↑	SSIM ↑	LPIPS ↓	W10 LPIPS ↓	P95 error ↓
FLUX.1-dev, $\mathcal{N} = 4$					
Raw forecast	20.874	0.768	0.118	0.251	0.330
Hard simplex	20.972	0.769	0.126	**0.246**	**0.317**
Fixed soft blend	20.970	0.767	0.118	0.250	0.327
Risk-adaptive blend	**21.122**	**0.772**	**0.116**	0.248	0.323
HunyuanVideo, $\mathcal{N} = 5$					
Raw forecast	26.178	0.946	0.146	0.362	0.116
Hard simplex	26.739	0.960	0.142	0.255	**0.100**
Fixed soft blend	26.920	**0.962**	**0.129**	0.258	0.114
Risk-adaptive blend	**26.959**	**0.962**	**0.129**	**0.250**	0.112

strengths are 0.2732 for FLUX and 0.1591 for HunyuanVideo; all other settings are held constant. Hard projection yields the lowest P95 output error on both models, but raises FLUX average LPIPS from 0.118 to 0.126. Risk-adaptive blending provides the best PSNR on both models and the lowest HunyuanVideo worst-10% LPIPS, balancing average fidelity with tail-error reduction. Activation and risk–error statistics appear in Appendix A.9.

Further design ablations. Appendix A.12 reports the regularization-strength and control-point sweeps, the perturbation-injection diagnostic, and the PCA-versus-simplex comparison. These results support the default cubic trajectory ($n = 4$), Tikhonov coefficient $\lambda = 0.5$, and coefficient-space guardrail.

5 Conclusion

We presented BeziCast, a training-free diffusion acceleration framework that forecasts output-proximal denoising representations with low-order Bézier trajectories. Across multiple image and video architectures, cache schedules, and sampling settings, BeziCast consistently improves the efficiencyreference-fidelity trade-off over competing cache-based methods. Matched-budget and mechanistic analyses further validate the local Bernstein forecaster and the risk-adaptive guardrail.

6 Acknowledgements

Jie Chen is the corresponding author. This work was supported in part by the New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0122702), the Shenzhen Medical Research Funds in China (No. B2302037), the National Natural Science Foundation of China under Grant U24B6012, 62406167, Natural Science Foundation of China (No. 61972217, 32071459, 62176249, 62006133, 62271465), AI for Science (AI4S)-Preferred Program, Peking University Shenzhen Graduate School, China, the Shenzhen Loop Area Institute under grant FPF10120250014, and Guangdong Provincial Key Laboratory of Artificial General Intelligence and Large Models (No. 2026B1212010005).

References

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

[2] Defang Chen, Zhenyu Zhou, Can Wang, and Siwei Lyu. Geometric regularity in deterministic sampling dynamics of diffusion-based generative models. *Journal of Statistical Mechanics: Theory and Experiment*, 2025(12):124002, 2025.

[3] Defang Chen, Zhenyu Zhou, Can Wang, Chunhua Shen, and Siwei Lyu. On the trajectory regularity of ODE-based diffusion sampling. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 7905–7934. PMLR, 21–27 Jul 2024.

[4] 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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12606–12633. PMLR, 21–27 Jul 2024.

[5] Rida Farouki and Tim Goodman. On the optimal stability of the bernstein basis. *Mathematics of computation*, 65(216):1553–1566, 1996.

[6] Rida T Farouki and VT Rajan. On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, 4(3):191–216, 1987.

[7] Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, et al. Hicache: A plug-in scaled-hermite upgrade for taylor-style cache-then-forecast diffusion acceleration. In *The Fourteenth International Conference on Learning Representations*, 2026.

[8] Jiaxi Gu, Shicong Wang, Haoyu Zhao, Tianyi Lu, Xing Zhang, Zuxuan Wu, Songcen Xu, Wei Zhang, Yu-Gang Jiang, and Hang Xu. Reuse and diffuse: Iterative denoising for text-to-video generation. *arXiv preprint arXiv:2309.03549*, 2023.

[9] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. LTX-Video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[10] Jiaqi Han, Juntong Shi, Puheng Li, Haotian Ye, Qiushan Guo, and Stefano Ermon. Adaptive spectral feature forecasting for diffusion sampling acceleration. *arXiv preprint arXiv:2603.01623*, 2026.

[11] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 7514–7528, 2021.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[13] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

[14] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[15] 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.

[16] Hao Li, He Cao, Bin Feng, Yanjun Shao, Xiangru Tang, Zhiyuan Yan, Li Yuan, Yonghong Tian, and Yu Li. Beyond chemical qa: Evaluating llm's chemical reasoning with modular chemical operations. *arXiv preprint arXiv:2505.21318*, 2025.

[17] Hao Li, Jinfa Huang, Peng Jin, Guoli Song, Qi Wu, and Jie Chen. Weakly-supervised 3d spatial reasoning for text-based visual question answering. *IEEE Transactions on Image Processing*, 32:3367–3382, 2023.

[18] Hao Li, Yanhao Jia, Peng Jin, Zesen Cheng, Kehan Li, Jialu Sui, Chang Liu, and Li Yuan. Freestyleret: retrieving images from style-diversified queries. In *European Conference on Computer Vision*, pages 258–274. Springer, 2024.

[19] 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 Computer Vision and Pattern Recognition Conference*, pages 7353–7363, 2025.

[20] Jiacheng Liu, Peiliang Cai, Qinming Zhou, Yuqi Lin, Deyang Kong, Benhao Huang, Yupei Pan, Haowen Xu, Chang Zou, Junshu Tang, et al. Freqca: Accelerating diffusion models via frequency-aware caching. *arXiv preprint arXiv:2510.08669*, 2025.

[21] 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*, pages 15853–15863, 2025.

[22] 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*, pages 10024–10033, 2025.

[23] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023.

[24] Yunfei Liu, Lei Zhu, Lijian Lin, Ye Zhu, Ailing Zhang, and Yu Li. TEASER: Token enhanced spatial modeling for expressions reconstruction. In *International Conference on Learning Representations*, 2025.

[25] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

[26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, 2025.

[27] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[28] 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*, pages 15762–15772, 2024.

[29] Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. Magcache: Fast video generation with magnitude-aware cache, 2025.

[30] Yunhong Min, Juil Koo, Seungwoo Yoo, and Minhyuk Sung. Bézierflow: Learning bézier stochastic interpolant schedulers for few-step generation, 2026.

[31] Nicholas M. Patrikalakis, Takashi Maekawa, and Wonjoon Cho. *Shape Interrogation for Computer Aided Design and Manufacturing*. 2009. Hyperbook edition.

[32] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

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

[34] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[35] 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.

[36] Zewen Shen and Kirill Serkh. On polynomial interpolation in the monomial basis. *SIAM Journal on Numerical Analysis*, 63(2):469494, March 2025.

[37] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[38] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 23–29 Jul 2023.

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

[40] Guangyi Wang, Yuren Cai, Lijiang Li, Wei Peng, and Songzhi Su. Pfdiff: Training-free acceleration of diffusion models combining past and future scores. *arXiv preprint arXiv:2408.08822*, 2024.

[41] Guangyi Wang, Wei Peng, Wenyu Chen, Yuren Cai, Song-Zhi Su, et al. Diffusion sampling correction via approximately 10 parameters. In *Forty-second International Conference on Machine Learning*, 2025.

[42] Jiahao Wu, Yunfei Liu, Lijian Lin, Ye Zhu, Lei Zhu, Jingyi Li, and Yu Li. PEAR: Pixel-aligned expressive human mesh recovery. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12. ACM, 2026.

[43] Quanwei Yang, Jiazhi Guan, Kaisiyuan Wang, Lingyun Yu, Wenqing Chu, Hang Zhou, Zhiqiang Feng, Haocheng Feng, Errui Ding, Jingdong Wang, and Hongtao Xie. ShowMaker: Creating high-fidelity 2d human video via fine-grained diffusion modeling. In *Advances in Neural Information Processing Systems*, volume 37, pages 51039–51062, 2024.

[44] Quanwei Yang, Luying Huang, Kaisiyuan Wang, Jiazhi Guan, Shengyi He, Fengguo Li, Hang Zhou, Lingyun Yu, Yingying Li, Haocheng Feng, and Hongtao Xie. GestureHYDRA: Semantic co-speech gesture synthesis via hybrid modality diffusion transformer and cascaded-synchronized retrieval-augmented generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12615–12625, 2025.

[45] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[46] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024.

[47] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.

[48] Dongxu Yue, Qin Guo, Munan Ning, Jiaxi Cui, Yuesheng Zhu, and Li Yuan. Chat-Face: Chat-guided real face editing via diffusion latent space manipulation. *arXiv preprint arXiv:2305.14742*, 2023.

[49] Dongxu Yue, Maomao Li, Yunfei Liu, Qin Guo, Ailing Zeng, Tianyu Yang, and Yu Li. AddMe: Zero-shot group-photo synthesis by inserting people into scenes. In *European Conference on Computer Vision*, pages 222–239. Springer, 2024.

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

[51] Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, et al. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13449–13457, 2026.

[52] Zhixin Zheng, Xinyu Wang, Chang Zou, Shaobo Wang, and Linfeng Zhang. Compute only 16 tokens in one timestep: Accelerating diffusion transformers with cluster-driven feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10181–10189, 2025.

[53] Zhenyu Zhou, Defang Chen, Siwei Lyu, Chun Chen, and Can Wang. Analyzing and improving fast sampling of text-to-image diffusion models. *arXiv preprint arXiv:2603.00763*, 2026.

[54] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7777–7786, 2024.

[55] Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. *Advances in Neural Information Processing Systems*, 37:40831–40860, 2024.

[56] Lei Zhu, Xing Cai, Yingjie Chen, Yiheng Li, Binxin Yang, Hao Liu, Jie Chen, Chen Li, and Jing Lyu. OmniHuman: A large-scale dataset and benchmark for human-centric video generation. In *European Conference on Computer Vision*, pages 92–110. Springer, 2026.

[57] Lei Zhu, Lijian Lin, Ye Zhu, Jiahao Wu, Xuehan Hou, Yu Li, Yunfei Liu, and Jie Chen. RealDyadic: Synthesizing realistic dyadic 3d dialogue with neural appearance priors. In *European Conference on Computer Vision*, pages 527–545. Springer, 2026.

[58] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction present BeziCast, a training-free acceleration method for diffusion, and the main contributions are supported by the method, theoretical analysis, and experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the appendix, including the methods dependence on the temporal regularity of output-proximal representations, the sensitivity of the speedquality trade-off to caching schedules and hyperparameters, and the need for further validation across more model architectures, samplers, resolutions, and long-video scenarios.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper states the assumptions for each theoretical statement, such as local Lipschitz conditions for output-proximal and sampler-level error propagation, and provides complete derivations or proofs in the main text and Appendix A. In particular, Appendix A gives formal proofs for the Taylor perturbation bound, ridge-stabilized Bernstein/Bézier stability, and output-proximal error propagation analysis.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides the main information needed to reproduce the experiments, including the evaluated models and tasks, benchmark protocols, baseline methods, caching intervals, key BeziCast hyperparameters, and evaluation metrics. Since BeziCast is training-free, no training data splits, optimizer, or training schedule are required; the experiments can be reproduced by applying the described inference-time cache-then-forecast procedure to the specified pretrained models and benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: the code is realeased at anonymous repo in the supplementrary materials.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the evaluated tasks, pretrained models, benchmarks, baselines, metrics, caching intervals, and key BeziCast hyperparameters such as the number of Bézier control points, Tikhonov regularization strength, warm-up length, cache period, history length, and guardrail settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper reports single-run results for the main comparisons and does not include error bars, confidence intervals, or statistical significance tests. However, the evaluation follows fixed benchmark protocols, and we provide extensive ablations and sensitivity analyses over key hyperparameters to assess the robustness of the observed speedquality trade-offs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the hardware used for evaluation, including the GPU type and memory, and provides end-to-end latency measurements for each method and task. These details, together with the specified pretrained models, benchmarks, resolutions, and inference settings, provide the necessary compute information to reproduce the reported experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses existing pretrained diffusion models and public evaluation benchmarks for studying training-free inference acceleration. It does not involve human subjects, private data collection, or the release of new generative model checkpoints, and we have reviewed the NeurIPS Code of Ethics to ensure compliance.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the positive impact of reducing diffusion inference cost, which can make high-quality image and video generation more efficient and accessible. It also acknowledges potential negative impacts, since faster generative models may lower the cost of misuse such as synthetic media generation, and emphasizes that the method does not introduce new model capabilities or release new generative checkpoints.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained model, generative checkpoint, scraped dataset, or other high-risk asset. The contribution is a training-free inference-time acceleration method evaluated on existing models and public benchmarks, so no additional release-specific safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper uses existing pretrained models, benchmarks, and baseline methods, and credits their creators through citations. We also follow the corresponding licenses and terms of use for all used assets, including pretrained diffusion models, evaluation benchmarks, and open-source baseline implementations.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces a new implementation of BeziCast. The method, inference algorithm, hyperparameters, and evaluation settings are documented in the main paper and appendix, and the released code will include instructions for reproducing the main experiments and applying the method to the evaluated diffusion models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments, user studies, or research with human subjects. Therefore, there are no participant instructions, screenshots, or compensation details to report.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects, user studies, crowdsourcing, or newly collected participant data. Therefore, there are no study-participant risks, risk disclosures, or IRB approval requirements associated with this work.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The core method does not use LLMs as an important, original, or non-standard component. Any use of LLMs for writing, editing, or formatting does not affect the methodology, experiments, or scientific conclusions.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

A Technical appendices and supplementary material

A.1 Why Use a Tikhonov-Stabilized Bernstein Basis?

A.1.1 Design Motivation of Tikhonov-Stabilized Bernstein Basis

The temporal basis determines how a cache-then-forecast method turns a short history of computed denoising representations into predictions at skipped timesteps [21, 7, 10]. Taylor-style predictors estimate local finite differences and extrapolate them in a local polynomial [21, 7]. This design is effective for short horizons, but it couples two sources of sensitivity: finite differences first form alternating high-order derivative estimates from nearby cached features, and extrapolation then multiplies these estimates by coefficients that grow with the forecast offset and expansion order. Proposition A.1 makes this mechanism explicit: for an order-p backward finite-difference predictor at future offset q, bounded cache perturbations are amplified by a factor upper bounded by $\sum_{j=0}^{p} \binom{q+j-1}{j} 2^j$ in Eq. (24), whose leading term grows as a degree-p polynomial in q.

BeziCast replaces direct reuse and high-order finite-difference extrapolation with a regularized local trajectory fit. It estimates a low-order Bernstein/Bézier curve from recent output-proximal representations and evaluates the fitted curve at cache-step coordinates. A Bézier curve is defined by a small set of control points and Bernstein basis functions [6]; once the control points are fixed, the curve can be evaluated analytically at any normalized time coordinate [31]. Thus, the number of control points sets trajectory capacity, while the cache schedule only determines how often the trajectory is queried.

Cache forecasting differs from standard Bézier interpolation because future full-step representations are unavailable during online inference. At a cache step, the method observes only previously computed full steps, so the query is often a short-range extrapolation beyond the most recent full step, i.e., $u > 1$. We therefore do not use the in-interval convex-hull property of Bézier curves as a direct correctness guarantee. Instead, we use the Bernstein/Bézier form as a smooth low-order trajectory family and explicitly control extrapolation through Tikhonov regularization, query-range clipping, and the equivalent-weight guardrail introduced in Section 3.3.

The Bernstein basis provides the geometric structure behind this design. On the standard interval $u \in [0, 1]$, Bernstein basis functions are nonnegative and form a partition of unity, so the corresponding Bézier curve lies in the convex hull of its control points [31]. Farouki and Goodman further show that the Bernstein basis has optimal stability among nonnegative polynomial bases on a fixed interval [5]. Although cache prediction may query $u > 1$, these properties remain useful as a diagnostic lens: after Tikhonov fitting, each prediction can be written as an equivalent linear combination of historical representations. If the equivalent weights become heavily signed or deviate from an affine combination, the forecast is relying on aggressive extrapolation rather than a conservative combination of observed features. This observation motivates both the regularized fitting below and the coefficient-space guardrail that follows [16, 18].

A.1.2 Perturbation Bounds and Proofs

This section formalizes the temporal-basis motivation in Section 3.2 and Appendix A.1.1. We first derive a perturbation bound for Taylor-style finite-difference extrapolation, and then show that the ridge-stabilized Bernstein/Bézier predictor admits a regularization-controlled stability bound.

Perturbation amplification in Taylor-style forecasting. Consider a scalar feature trajectory observed at equally spaced historical full steps,

$$z_0, z_{-1}, \ldots, z_{-p}. \tag{22}$$

Here $p \geq 0$ is an integer and consecutive observations are one time unit apart. For future offset $q > 0$, the degree-p Newton backward extrapolant is

$$\hat{z}_{\mathrm{Tay}}(q) = \sum_{j=0}^{p} \binom{q+j-1}{j} \nabla^j z_0, \tag{23}$$

where ∇^j is the j-th backward difference, with $\nabla^0 z_0 = z_0$, and $\binom{q+j-1}{j} = q(q+1)\cdots(q+j-1)/j!$, with the $j = 0$ coefficient equal to one. The rising-factorial coefficient corresponds to

backward differences; the falling-factorial coefficient $\binom{q}{j}$ belongs to the forward-difference form. This extrapolant illustrates the finite-difference mechanism rather than specifying every Taylor-style implementation.

Proposition A.1 (backward-difference perturbation). If each cached value is perturbed by η_{-r} with $|\eta_{-r}| \le \varepsilon$ for $r = 0, \dots, p$, where $\varepsilon \ge 0$, then the induced perturbation of Eq. (23) satisfies

$$|\delta\hat{z}_{\mathrm{Tay}}(q)| \le \varepsilon \sum_{j=0}^{p} \binom{q+j-1}{j} 2^j. \tag{24}$$

For fixed order p, the leading factor in this upper bound grows as a degree-p polynomial in the forecast offset q.

Proof. The j-th backward difference can be expanded as

$$\nabla^j z_0 = \sum_{r=0}^{j} (-1)^r \binom{j}{r} z_{-r}, \tag{25}$$

so Eq. (23) can be written as a linear extrapolation over historical values,

$$\hat{z}_{\mathrm{Tay}}(q) = \sum_{r=0}^{p} w_r^{\mathrm{Tay}}(q) z_{-r}. \tag{26}$$

Here $w_r^{\mathrm{Tay}}(q) = (-1)^r \sum_{j=r}^{p} \binom{q+j-1}{j}\binom{j}{r}$ is the coefficient of z_{-r}. By linearity, the perturbation is $\delta\hat{z}_{\mathrm{Tay}}(q) = \sum_{r=0}^{p} w_r^{\mathrm{Tay}}(q)\eta_{-r}$. Therefore,

$$|\delta\hat{z}_{\mathrm{Tay}}(q)| \le \Lambda_p^{\mathrm{Tay}}(q)\varepsilon, \qquad \Lambda_p^{\mathrm{Tay}}(q) = \sum_{r=0}^{p} |w_r^{\mathrm{Tay}}(q)|. \tag{27}$$

Using the triangle inequality before collecting the coefficients gives

$$\Lambda_p^{\mathrm{Tay}}(q) \le \sum_{j=0}^{p} \binom{q+j-1}{j} 2^j. \tag{28}$$

The asymptotic statement follows because $\binom{q+p-1}{p}$ is a degree-p polynomial in q. For vector observations, the same triangle-inequality argument holds with Euclidean norms and $\|\eta_{-r}\|_2 \le \varepsilon$.

This proposition identifies a perturbation mechanism rather than a universal failure mode. Taylor-style predictors can work well for short extrapolation offsets or low expansion orders, but the finite-difference form allows small cache errors to be magnified when the cache interval is aggressive.

Stabilized Bernstein/Bézier forecasting. BeziCast instead estimates a low-order Bernstein/Bézier trajectory from cached output-proximal representations. Let

$$Z = \begin{bmatrix} z_{i_1} \\ \vdots \\ z_{i_K} \end{bmatrix} \in \mathbb{R}^{K \times D} \tag{29}$$

stack the K cached row vectors, as in the main method. For matrices, $\|\cdot\|_2$ denotes the spectral norm and $\|\cdot\|_F$ the Frobenius norm. Let n be the number of Bézier control representations, so the polynomial degree is $n-1$. For normalized coordinate u, define the Bernstein basis vector $b(u) \in \mathbb{R}^n$ by

$$b_{j,n-1}(u) = \binom{n-1}{j} u^j (1-u)^{n-1-j}, \qquad j = 0, \dots, n-1, \tag{30}$$

$$b(u) = [b_{0,n-1}(u), \dots, b_{n-1,n-1}(u)]^\top \in \mathbb{R}^n. \tag{31}$$

Let the Bernstein design matrix $A \in \mathbb{R}^{K \times n}$ be

$$A_{m,j+1} = b_{j,n-1}(u_m^{\mathrm{hist}}), \qquad m = 1, \dots, K, \quad j = 0, \dots, n-1 \tag{32}$$

for historical coordinates $u_1^{\text{hist}}, \ldots, u_K^{\text{hist}}$. The control matrix $P \in \mathbb{R}^{n \times D}$ is estimated by Tikhonov-regularized least squares,

$$P_\lambda^\star = \arg\min_P \|AP - Z\|_F^2 + \lambda \|\Gamma P\|_F^2, \tag{33}$$

where $\lambda > 0$ controls the regularization strength and Γ is the Tikhonov matrix with n columns and a compatible number of rows. If $A^\top A + \lambda \Gamma^\top \Gamma$ is invertible, the closed-form solution of Eq. (33) is

$$P_\lambda^\star = \left(A^\top A + \lambda \Gamma^\top \Gamma\right)^{-1} A^\top Z. \tag{34}$$

In the main implementation, we use $\Gamma = I$, yielding the ridge-stabilized solution

$$P_\lambda^\star = (A^\top A + \lambda I)^{-1} A^\top Z. \tag{35}$$

Since $A^\top A + \lambda I$ is positive definite for $\lambda > 0$, this estimator remains well posed even when A is rank-deficient or ill-conditioned. The prediction at query coordinate u is

$$\hat{z}_{\text{Bez}}(u) = b(u)^\top P_\lambda^\star = b(u)^\top (A^\top A + \lambda I)^{-1} A^\top Z. \tag{36}$$

Proposition A.2 (ridge Bernstein stability). Fix A, u, and $\lambda > 0$. For the raw predictor in Eq. (36), any cache perturbation δZ induces

$$\|\delta \hat{z}_{\text{Bez}}(u)\|_2 \le \frac{\|b(u)\|_2}{2\sqrt{\lambda}} \|\delta Z\|_F. \tag{37}$$

Moreover, for the clipped extrapolation range $u \in [0, u_{\max}]$, define

$$C_b(u_{\max}, n) = \sup_{u \in [0, u_{\max}]} \|b(u)\|_2. \tag{38}$$

Then

$$\|\delta \hat{z}_{\text{Bez}}(u)\|_2 \le \frac{C_b(u_{\max}, n)}{2\sqrt{\lambda}} \|\delta Z\|_F, \qquad u \in [0, u_{\max}]. \tag{39}$$

Proof. Because Eq. (36) is linear in Z,

$$\delta \hat{z}_{\text{Bez}}(u) = b(u)^\top (A^\top A + \lambda I)^{-1} A^\top \delta Z. \tag{40}$$

Hence

$$\|\delta \hat{z}_{\text{Bez}}(u)\|_2 \le \|b(u)\|_2 \left\|(A^\top A + \lambda I)^{-1} A^\top\right\|_2 \|\delta Z\|_F. \tag{41}$$

Let $A = U \Sigma V^\top$ be a singular value decomposition, with singular values $\sigma_i \ge 0$ on the diagonal of Σ and orthogonal U, V. Then

$$\left\|(A^\top A + \lambda I)^{-1} A^\top\right\|_2 = \max_i \frac{\sigma_i}{\sigma_i^2 + \lambda} \le \frac{1}{2\sqrt{\lambda}}, \qquad \lambda > 0. \tag{42}$$

This proves Eq. (37); taking the supremum of $\|b(u)\|_2$ over $u \in [0, u_{\max}]$ gives Eq. (39).

The bound separates a regularization-controlled factor from the query factor $\|b(u)\|_2$. The latter still grows polynomially outside $[0, 1]$ and depends on degree; clipping makes it uniformly bounded on the allowed query range. This controls sensitivity to cache perturbations at fixed coordinates, while approximation bias and coordinate changes are separate error sources.

Why not ordinary least squares? The preceding bound depends on the Tikhonov term. Without this term, the Bernstein control points are estimated by ordinary least squares (OLS),

$$P_{\text{OLS}}^\star = A^+ Z, \tag{43}$$

where A^+ is the Moore–Penrose pseudoinverse; this selects the minimum-norm solution if the least-squares minimizer is not unique. When A has full column rank, Eq. (43) reduces to

$$P_{\text{OLS}}^\star = (A^\top A)^{-1} A^\top Z. \tag{44}$$

The induced prediction at query coordinate u is

$$\hat{z}_{\text{OLS}}(u) = b(u)^\top A^+ Z. \tag{45}$$

Proposition A.3 (OLS versus ridge sensitivity). Fix the Bernstein design matrix A and query coordinate u. Let $A = U\Sigma V^\top$ have rank $r \geq 1$ and nonzero singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$. For any cache perturbation δZ, the OLS predictor satisfies

$$\|\delta\hat{z}_{\mathrm{OLS}}(u)\|_2 \leq \frac{\|b(u)\|_2}{\sigma_r}\|\delta Z\|_F, \tag{46}$$

whereas the ridge-stabilized predictor $\hat{z}_\lambda = \hat{z}_{\mathrm{Bez}}$ with $\lambda > 0$ satisfies

$$\|\delta\hat{z}_\lambda(u)\|_2 \leq \frac{\|b(u)\|_2}{2\sqrt{\lambda}}\|\delta Z\|_F. \tag{47}$$

Proof. For OLS, linearity gives

$$\delta\hat{z}_{\mathrm{OLS}}(u) = b(u)^\top A^+ \delta Z. \tag{48}$$

Using the spectral/Frobenius norm inequality,

$$\|\delta\hat{z}_{\mathrm{OLS}}(u)\|_2 \leq \|b(u)\|_2\|A^+\|_2\|\delta Z\|_F. \tag{49}$$

Since $\|A^+\|_2 = 1/\sigma_r$, Eq. (46) follows. For the ridge estimator,

$$P^\star_\lambda = (A^\top A + \lambda I)^{-1}A^\top Z, \tag{50}$$

and the same argument yields

$$\|\delta\hat{z}_\lambda(u)\|_2 \leq \|b(u)\|_2 \left\|(A^\top A + \lambda I)^{-1}A^\top\right\|_2 \|\delta Z\|_F. \tag{51}$$

In the singular-vector basis, the spectral filter of $(A^\top A + \lambda I)^{-1}A^\top$ is

$$g_\lambda(\sigma_i) = \frac{\sigma_i}{\sigma_i^2 + \lambda}. \tag{52}$$

Therefore,

$$\left\|(A^\top A + \lambda I)^{-1}A^\top\right\|_2 = \max_i g_\lambda(\sigma_i) \leq \sup_{\sigma\geq 0}\frac{\sigma}{\sigma^2+\lambda} = \frac{1}{2\sqrt{\lambda}}, \tag{53}$$

which proves Eq. (47).

This proposition isolates the role of Tikhonov stabilization. The OLS sensitivity depends on $1/\sigma_r$, which can be large when the Bernstein design matrix is ill-conditioned or nearly rank-deficient. By contrast, the ridge predictor replaces this data-dependent factor with the explicit regularization-controlled factor $1/(2\sqrt{\lambda})$. In each nonzero singular direction, OLS applies the filter $1/\sigma_i$, while ridge applies $\sigma_i/(\sigma_i^2 + \lambda)$. Their ratio is

$$\frac{\sigma_i/(\sigma_i^2+\lambda)}{1/\sigma_i} = \frac{\sigma_i^2}{\sigma_i^2+\lambda} \leq 1, \tag{54}$$

so ridge attenuates directions that are weakly observed by the cache design. Thus, Tikhonov fitting preserves the Bernstein/Bézier trajectory class but stabilizes the estimation of its control representations.

This stability is important for training-free cache-step forecasting. The history window contains only a few full denoiser evaluations, and cache queries are often extrapolative ($u > 1$). Conditioning is determined by the historical coordinates and basis in A, not by correlations among the rows of Z. In this regime, OLS may fit small cache perturbations or local fluctuations too closely, and these perturbations can be propagated by extrapolative Bernstein weights. Tikhonov fitting instead trades exact interpolation of the cached history for a smoother and better-conditioned control-point estimate.

The same trade-off can be expressed by a bias–perturbation decomposition. Suppose the cached representations follow a local model

$$Z = AP^\star + E, \tag{55}$$

where $P^\star$ is an ideal local control-point matrix and E denotes cache noise or approximation error. If A has full column rank, then

$$P^\star_{\mathrm{OLS}} - P^\star = A^+E, \tag{56}$$

whereas ridge gives

$$P^\star_\lambda - P^\star = -\lambda(A^\top A + \lambda I)^{-1}P^\star + (A^\top A + \lambda I)^{-1}A^\top E. \tag{57}$$

The first term is deterministic shrinkage and the second is the perturbation response. When E is zero-mean noise, these are the statistical bias and zero-mean estimation error, respectively. The Tikhonov term damps the small-singular-value components of the noise term at the cost of this controlled bias. Since the goal of BeziCast is stable extrapolation at skipped steps rather than exact reconstruction of the cached history, this trade-off is better aligned with the inference-time forecasting objective.

A.2 Why Forecast Output-Proximal Representations?

This subsection explains why BeziCast forecasts an output-proximal representation z_s rather than intermediate module outputs. We first give a local error-propagation analysis. We then report ablations on the forecasting target, perturbation sensitivity, and the step at which output-proximal forecasting is activated.

A.2.1 Theoretical Analysis

The key distinction is the error-injection point. An intermediate prediction error must pass through the remaining denoiser blocks before reaching the final noise estimate, whereas an output-proximal error only passes through the final output map.

Intermediate-module error propagation. Let h_s^ℓ be the representation at layer ℓ, and let L index the final hidden representation $h_s^L = z_s$. With t_s, c fixed, denote the downstream block maps by Φ_r:

$$h_s^{r+1} = \Phi_r(h_s^r, t_s, c), \qquad r = \ell, \dots, L-1, \tag{58}$$

with final denoising output

$$\epsilon_s = \mathcal{O}_\theta(h_s^L, t_s, c). \tag{59}$$

Proposition A.4 (intermediate-module error propagation). Suppose the downstream blocks $\Phi_\ell, \dots, \Phi_{L-1}$ have local Lipschitz constants $L_\ell, \dots, L_{L-1}$, and the output map $\mathcal{O}_\theta$ has local Lipschitz constant $L_\mathcal{O}$, on neighborhoods containing both the exact and perturbed paths. If the intermediate representation is replaced by

$$\hat{h}_s^\ell = h_s^\ell + e_s^\ell, \tag{60}$$

then the induced denoising-output difference $\delta\epsilon_s$ satisfies

$$\|\delta\epsilon_s\| \le L_\mathcal{O} \prod_{r=\ell}^{L-1} L_r \|e_s^\ell\|. \tag{61}$$

Proof. The denoising error caused by replacing h_s^ℓ with $\hat{h}_s^\ell$ is

$$\begin{aligned} \delta\epsilon_s = \mathcal{O}_\theta \circ \Phi_{L-1} \circ \cdots \circ \Phi_\ell(h_s^\ell + e_s^\ell) \\ - \mathcal{O}_\theta \circ \Phi_{L-1} \circ \cdots \circ \Phi_\ell(h_s^\ell). \end{aligned} \tag{62}$$

Here the fixed arguments t_s, c are suppressed in each map. Applying the local Lipschitz bounds sequentially through the composed downstream map gives Eq. (61).

The product of downstream constants can be above or below one: intermediate errors may be amplified or contracted along this path. Under the same local Lipschitz assumptions, a corresponding bound for multiple predicted modules can be obtained by accumulating their downstream-modulated terms.

Output-proximal error propagation. The output-proximal representation z_s in Eq. (2) is placed immediately before the final output map:

$$\epsilon_s = \mathcal{O}_\theta(z_s, t_s, c). \tag{63}$$

Proposition A.5 (output-proximal error propagation). Assume the output map $\mathcal{O}_\theta$ has local Lipschitz constant $L_\mathcal{O}$ on a neighborhood containing the exact and predicted representation inputs. If

$$\hat{z}_s = z_s + e_s^z, \tag{64}$$

then

$$\|\mathcal{O}_\theta(\hat{z}_s, t_s, c) - \mathcal{O}_\theta(z_s, t_s, c)\| \le L_{\mathcal{O}} \|e_s^z\|. \tag{65}$$

Proof. Equation (65) follows directly from the local Lipschitz assumption on $\mathcal{O}_\theta$.

Compared with Eq. (61), Eq. (65) removes the additional product of downstream block constants. This does not imply that z_s is always easier to forecast than intermediate activations. Rather, it makes the relationship between representation prediction error and denoising-output error more direct, which is the property used by BeziCast.

Sampler-level consequence. To separate local forecasting error from trajectory drift, let x_s and $\hat{x}_s$ denote the full and accelerated trajectories, respectively. Write $\epsilon_s = \epsilon_\theta(x_s, t_s, c)$, and let $\hat{\epsilon}_s$ be the denoising output actually used on the accelerated trajectory. Both use

$$x_{s+1} = \Psi_s(x_s, \epsilon_s), \qquad \hat{x}_{s+1} = \Psi_s(\hat{x}_s, \hat{\epsilon}_s). \tag{66}$$

Define the same-state approximation error $e_s^\epsilon = \hat{\epsilon}_s - \epsilon_\theta(\hat{x}_s, t_s, c)$. It is zero at full steps; at cache steps it isolates forecasting error from the effect of evaluating the denoiser at a different latent state.

Proposition A.6 (sampler-level error propagation). Assume Ψ_s has state and output Lipschitz constants L_s^x, L_s^ϵ on neighborhoods containing both trajectories:

$$\|\Psi_s(x, \epsilon) - \Psi_s(x', \epsilon')\| \le L_s^x \|x - x'\| + L_s^\epsilon \|\epsilon - \epsilon'\|. \tag{67}$$

Assume also that $\epsilon_\theta(\cdot, t_s, c)$ has Lipschitz constant L_s^θ on these states, and define $a_s = L_s^x + L_s^\epsilon L_s^\theta$. If $\hat{x}_0 = x_0$, then

$$\|\hat{x}_T - x_T\| \le \sum_{s=0}^{T-1} \left(L_s^\epsilon \prod_{r=s+1}^{T-1} a_r \right) \|e_s^\epsilon\|. \tag{68}$$

An empty product equals one. Stochastic updates are compared with identical random draws, so the bound is conditional on those draws.

Proof. Let $d_s = \|\hat{x}_s - x_s\|$. The triangle inequality gives

$$\|\hat{\epsilon}_s - \epsilon_s\| \le \|e_s^\epsilon\| + L_s^\theta d_s. \tag{69}$$

Combining this with the sampler Lipschitz bound yields $d_{s+1} \le a_s d_s + L_s^\epsilon \|e_s^\epsilon\|$. Unrolling from $d_0 = 0$ yields Eq. (68).

At a cache step, Proposition A.5 gives $\|e_s^\epsilon\| \le L_{\mathcal{O}} \|\tilde{z}_s - \mathcal{F}_\theta(\hat{x}_s, t_s, c)\|$. This connects the same-state representation error to accumulated trajectory error; the factors a_s account for feedback through both the sampler and the denoiser.

A.2.2 Ablation: Forecasting Target

We first ablate the representation used as the forecasting target. To isolate the target choice, we keep the pretrained FLUX denoiser, sampler, cache schedule, history length, Bernstein order, Tikhonov coefficient, and guardrail configuration fixed, and vary only the predicted representation.

We compare three target types. **Block-wise forecasting** applies the same Bézier forecaster to local transformer-block outputs, resembling local-module feature caching where prediction errors are injected before subsequent blocks. **Residual forecasting** predicts the block residual/update representation, following the residual-style target used in prior caching methods. In our implementation, the residual is defined as the difference between the block output and the block input. **Output-proximal forecasting** predicts the final hidden representation z_s immediately before the output map $\mathcal{O}_\theta$, which is the default design of BeziCast.

Table 4 shows that output-proximal forecasting gives the best latency while matching the reference fidelity of residual forecasting. It has nearly identical PSNR, SSIM, and LPIPS to residual forecasting, but reduces latency from 14.016s to 11.530s and gives the best CLIP and ImageReward scores. Compared with block-wise forecasting, it improves all listed quality metrics and also reduces latency. These results indicate that the final hidden representation remains predictable enough for cache-step forecasting, while avoiding the cost of predicting multiple intermediate targets.

Table 4: Ablation on forecasting targets on FLUX. All variants use the same pretrained denoiser, sampler, cache schedule, history length, Bernstein order, Tikhonov coefficient, and guardrail configuration.

Forecasting target	PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	ImageReward ↑	Latency ↓
Block-wise forecasting	22.993	0.793	0.075	27.091	1.035	15.208
Residual forecasting	**23.289**	0.798	0.074	27.097	1.026	14.016
Output-proximal forecasting z_s	23.287	**0.798**	**0.074**	**27.099**	**1.038**	**11.530**

Table 5: Ablation on when to switch to output-proximal forecasting. Steps are 0-indexed; the first 8 steps are always full denoiser evaluations. Switching at the first cache step gives the best latency and overall fidelity in this setting.

Output-proximal switch step	PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	ImageReward ↑	Latency ↓
25	23.199	0.797	0.074	27.100	1.031	13.146
20	23.229	0.797	0.074	**27.111**	1.030	12.622
15	23.243	0.797	0.075	27.101	1.027	12.176
10	23.274	**0.798**	0.075	27.081	1.026	11.749
0 / before first cache step	**23.287**	**0.798**	**0.074**	27.099	**1.038**	**11.530**

Perturbation injection diagnostic. We next measure how a fixed relative representation error affects the final denoising output. For a candidate target q with representation h_s^q, let $\mathcal{G}_q$ denote the downstream computation from that representation to the denoising output. We inject a random perturbation with relative magnitude σ:

$$\tilde{h}_s^q = h_s^q + \sigma \frac{\xi}{\|\xi\|_2} \|h_s^q\|_2,$$

where $\xi \neq 0$ is random noise of the same shape as h_s^q and $\sigma = 0.01$. We then continue the forward pass from $\tilde{h}_s^q$ and measure

$$\Delta_\epsilon(q) = \frac{\|\mathcal{G}_q(\tilde{h}_s^q) - \mathcal{G}_q(h_s^q)\|_2}{\|\mathcal{G}_q(h_s^q)\|_2}.$$

The denominator is assumed nonzero; the comparison fixes the timestep, conditioning, and all other activations. A smaller $\Delta_\epsilon(q)$ means that the same relative target error induces a smaller final-output deviation. Figure 6(b) displays this sensitivity as a robustness score $-\log \Delta_\epsilon(q)$, so larger scores indicate smaller output deviations.

We compare the final hidden state with double-stream block residuals and double-stream block outputs at layers 6, 10, 14, and 18. We focus on double-stream blocks only, to avoid mixing structurally different block types. Figure 6(b) shows that the final hidden state has the smallest output deviation under the same relative perturbation. Block residuals are less sensitive than block outputs at the same layer, and later blocks are generally less sensitive than earlier ones, but all probed block-level targets remain more sensitive than the final hidden representation. This matches the analysis above: intermediate errors pass through additional downstream computation, while the output-proximal target has the shortest path to the denoising output.

When to switch to output-proximal forecasting. Finally, we test whether output-proximal forecasting should start immediately after warm-up or only in later denoising steps. The first 8 solver steps are full denoiser evaluations in all variants. Before the switch step, cache steps use the block-wise target; from the switch step onward, cache steps use the output-proximal target z_s. All variants use the same cache schedule, $\lambda = 5 \times 10^{-1}$, and $n = 6$.

Table 5 shows that activating output-proximal forecasting from the beginning of the cache phase gives the lowest latency and the strongest overall fidelity among the tested switch points. Delaying the switch does not improve PSNR, SSIM, or LPIPS, and it increases latency because more cache steps use the more expensive block-wise target. This result is consistent with the target ablation: after a short full-step warm-up, the output-proximal representation is already a suitable forecasting target. We therefore use output-proximal forecasting from the first cache step in the main experiments.

A.3 Warm-up and Fixed-period Full-step Schedule

This subsection first defines the deterministic full-step schedule used by BeziCast, and then reports an ablation on the warm-up length. The schedule has two roles: the initial warm-up provides real output-proximal representations for the first local fit, and the later fixed-period full steps refresh the history used by cache-step forecasting.

A.3.1 Method Details

BeziCast uses a warm-up plus fixed-period full-step schedule. The warm-up length W is the number of initial solver steps evaluated by the full denoiser, and the fixed period R specifies how often a full denoiser evaluation is inserted after warm-up. At full steps, BeziCast computes and caches the true output-proximal representation; at cache steps, it predicts the current representation from the most recent cached full-step representations.

With n control representations, a determined or overdetermined unregularized fit requires n distinct history coordinates. A sufficient initialization condition is

$$W \geq n, \qquad K_{\max} \geq n. \tag{70}$$

In the main configuration, BeziCast uses a cubic Bernstein/Bézier forecaster with $n = 4$, matching the control-representation notation used in the experimental tables:

$$\begin{aligned} b_{0,3}(u) &= (1-u)^3, \qquad b_{1,3}(u) = 3u(1-u)^2, \\ b_{2,3}(u) &= 3u^2(1-u), \quad b_{3,3}(u) = u^3. \end{aligned} \tag{71}$$

Thus $W \geq 4$ and $K_{\max} \geq 4$ provide a data-determined cubic fit. With $\lambda > 0$, the ridge system remains uniquely solvable for $K < n$. The $W = 3$ ablation therefore starts with an underdetermined data fit stabilized by ridge, rather than requiring an additional full step.

After warm-up, full denoiser evaluations follow a fixed period. We use the same zero-indexed solver convention as the main method, $s \in \{0, \ldots, T-1\}$. Let the integer $R \geq 1$ be the full-step period after warm-up (denoted $\mathcal{N}$ in the experiments), meaning that one full denoiser evaluation is performed every R solver steps. The full-step set is

$$\mathcal{S}_{\text{full}} = \{0, \ldots, W-1\} \cup \{s \in \{W, \ldots, T-1\} : (s - W + 1) \equiv 0 \pmod{R}\}, \tag{72}$$

and the cache-step set is

$$\mathcal{S}_{\text{cache}} = \{0, \ldots, T-1\} \setminus \mathcal{S}_{\text{full}}. \tag{73}$$

For integer $2 \leq W \leq T$, this schedule makes

$$|\mathcal{S}_{\text{full}}| = W + \left\lfloor \frac{T-W}{R} \right\rfloor \tag{74}$$

full denoiser evaluations. The last warm-up step $W-1$ anchors the periodic refreshes, whose first occurrence is $W + R - 1$.

At a cache step s, the predictor uses only the most recent full-step representations:

$$\mathcal{I}_s = \text{Last}_{K_{\max}}\left(\{i \in \mathcal{S}_{\text{full}} : i < s\}\right), \tag{75}$$

where $\text{Last}_{K_{\max}}$ selects the largest at most $K_{\max}$ indices, in chronological order. The available length is $K_s = |\mathcal{I}_s|$, abbreviated K in each local fit. Unless otherwise stated, we set $K_{\max} = 6$. Therefore, each cache prediction uses at most six real output-proximal representations, and the cubic Bézier trajectory is fitted within a short local history window.

A.3.2 Ablation Experiments

We evaluate the warm-up schedule on FLUX.1-dev using the same DrawBench protocol and metrics as Section 4. All variants use the output-proximal target with $\lambda = 0.5$, $n = 4$, $K_{\max} = 6$, and fixed period $R = 6$. We vary only the warm-up length $W \in \{3, 4, 5, 6, 7, 8\}$. This isolates the effect of the number of initial full denoiser evaluations before the first cache prediction.

Table 6 shows a clear speed–fidelity trade-off. Increasing W gives the forecaster a longer real history before cache-step prediction begins and improves reference fidelity: PSNR increases from 17.070 at

Table 6: Warm-up schedule ablation on FLUX. All rows use output-proximal BeziCast with $R = 6$, $n = 4$, $K_{\max} = 6$, and $\lambda = 0.5$.

Warm-up W	PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	ImageReward ↑	Latency / Speedup
3	17.070	0.623	0.242	27.117	0.984	7.881 / 4.78×
4	18.598	0.678	0.180	27.310	1.045	8.732 / 4.32×
5	20.010	0.724	0.135	27.414	1.046	9.383 / 4.02×
6	20.993	0.750	0.115	27.192	1.028	10.109 / 3.73×
7	22.206	0.780	0.093	27.060	1.028	10.848 / 3.47×
8	23.287	0.798	0.074	27.099	1.038	11.530 / 3.27×

$W = 3$ to 23.287 at $W = 8$, SSIM increases from 0.623 to 0.798, and LPIPS decreases from 0.242 to 0.074. The cost is higher latency, which increases from 7.881s to 11.530s and reduces speedup from $4.78\times$ to $3.27\times$. CLIP and ImageReward vary in a narrower range, so this ablation mainly reflects the trade-off between reference fidelity and runtime.

We use the aggressive $W = 3$ schedule for the $\mathcal{N} = 6$ FLUX main result (in Table 1) because it provides the largest speedup in this ablation while still improving the matched TaylorSeer baselines. Specifically, under the same $\mathcal{N} = 6$ setting, BeziCast with $W = 3$ reaches 17.070 PSNR, 0.623 SSIM, and 0.242 LPIPS at 7.881s, while the best TaylorSeer row reports 15.872 PSNR, 0.621 SSIM, and 0.281 LPIPS at 8.862s. When higher reference fidelity is preferred over maximum speed, increasing W offers a direct schedule-level adjustment without changing the forecaster, output target, or pretrained denoiser.

A.4 Online Inference Algorithm

Algorithm 1 summarizes inference. We first construct the warm-up plus fixed-period full-step set. Full steps run the original denoiser and update the cache. Cache steps compute the equivalent ridge-Bézier weights, optionally blend with the convex-hull fallback, and pass the predicted output-proximal representation through the unchanged output map. We require $\min(W, K_{\max}) \geq 2$ to initialize a nondegenerate local time window. Ridge fitting is used even when $K_s < n$, so the full-step locations and NFE count remain exactly those of the chosen schedule.

A.5 Basis Dependence of Tikhonov-Regularized Forecasting

We clarify why two polynomial bases spanning the same finite-dimensional function space can nevertheless induce different regularized forecasters.

Let

$$\boldsymbol{\phi}(u) = [\phi_0(u), \ldots, \phi_{n-1}(u)]^\top \quad \text{and} \quad \boldsymbol{\psi}(u) = [\psi_0(u), \ldots, \psi_{n-1}(u)]^\top$$

be two bases of $\mathcal{P}_{n-1}$, the real univariate polynomials of degree at most $n-1$. Because both are bases of $\mathcal{P}_{n-1}$, there exists an invertible matrix $M \in \mathbb{R}^{n\times n}$ such that

$$\boldsymbol{\psi}(u) = M^\top \boldsymbol{\phi}(u). \tag{76}$$

For historical coordinates $\{u_m^{\text{hist}}\}_{m=1}^K$, define the corresponding $K \times n$ design matrices

$$A_\phi = \begin{bmatrix} \boldsymbol{\phi}(u_1^{\text{hist}})^\top \\ \vdots \\ \boldsymbol{\phi}(u_K^{\text{hist}})^\top \end{bmatrix}, \qquad A_\psi = \begin{bmatrix} \boldsymbol{\psi}(u_1^{\text{hist}})^\top \\ \vdots \\ \boldsymbol{\psi}(u_K^{\text{hist}})^\top \end{bmatrix}. \tag{77}$$

Equation (76) implies

$$A_\psi = A_\phi M. \tag{78}$$

Proposition A.7 (Basis invariance without regularization)**.** *Assume that A_ϕ has full column rank. Let $Z \in \mathbb{R}^{K\times D}$ be the matrix of cached representations. For $\lambda = 0$, the least-squares predictors constructed in the two bases are identical at every query coordinate u:*

$$\begin{aligned} &\boldsymbol{\phi}(u)^\top (A_\phi^\top A_\phi)^{-1} A_\phi^\top Z \\ &\quad = \boldsymbol{\psi}(u)^\top (A_\psi^\top A_\psi)^{-1} A_\psi^\top Z. \end{aligned} \tag{79}$$

Algorithm 1 BeziCast inference

Require: initial latent x_0, pretrained denoiser $(\mathcal{F}_\theta, \mathcal{O}_\theta)$, sampler $\{\Psi_s\}_{s=0}^{T-1}$, time grid $\{t_s\}_{s=0}^{T-1}$, strictly increasing coordinates $\{\tau_s\}_{s=0}^{T-1}$, condition c, warm-up length W, full-step period R, history cap $K_{\max}$, control-point number n, ridge strength $\lambda > 0$, normalization constant $\varepsilon_\tau > 0$, query cap $u_{\max}$, and guardrail parameters $(\beta_{\max}, \rho, \gamma)$.

Construct $\mathcal{S}_{\text{full}}$ by Eq. (72).
Initialize cache $\mathcal{C} \leftarrow \emptyset$.
for $s = 0, \dots, T-1$ **do**
  if $s \in \mathcal{S}_{\text{full}}$ **then**
    Compute $z_s = \mathcal{F}_\theta(x_s, t_s, c)$.
    Compute $\epsilon_s = \mathcal{O}_\theta(z_s, t_s, c)$.
    Update $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\tau_s, z_s)\}$ and keep the most recent $K_{\max}$ full-step entries.
  else
    Set $K_s = |\mathcal{C}|$ and retrieve $\{(\tau_{i_m}, z_{i_m})\}_{m=1}^{K_s}$ in time order.
    Compute $\{u_m^{\text{hist}}\}_{m=1}^{K_s}$ and u_s using Eqs. (6)–(7).
    Construct $A \in \mathbb{R}^{K_s \times n}$ and $Z \in \mathbb{R}^{K_s \times D}$.
    Compute $w = \omega_\lambda(u_s)$ using Eq. (13), and $\hat{z}_s = w^\top Z$.
    Compute r_s and β_s using Eqs. (15) and (21).
    Set $\tilde{z}_s = \hat{z}_s$.
    if $\beta_s > 0$ **then**
      Project $w_{\text{safe}} = \Pi_\Delta(w)$ and compute $z_s^{\text{safe}} = w_{\text{safe}}^\top Z$.
      Set $\tilde{z}_s = (1 - \beta_s)\hat{z}_s + \beta_s z_s^{\text{safe}}$.
    end if
    Compute $\hat{\epsilon}_s = \mathcal{O}_\theta(\tilde{z}_s, t_s, c)$.
  end if
  Update $x_{s+1} = \Psi_s(x_s, \epsilon_s)$ at full steps and $x_{s+1} = \Psi_s(x_s, \hat{\epsilon}_s)$ at cache steps.
end for
return x_T.

Proof. Let $G = A_\phi^\top A_\phi$. By Eq. (76) and Eq. (78),

$$\boldsymbol{\psi}(u)^\top = \boldsymbol{\phi}(u)^\top M, \qquad A_\psi^\top A_\psi = M^\top G M.$$

Since M is invertible,

$$(M^\top G M)^{-1} = M^{-1} G^{-1} M^{-\top}.$$

Therefore,

$$\begin{aligned}
&\boldsymbol{\psi}(u)^\top (A_\psi^\top A_\psi)^{-1} A_\psi^\top Z \\
&= \boldsymbol{\phi}(u)^\top M (M^\top G M)^{-1} M^\top A_\phi^\top Z \\
&= \boldsymbol{\phi}(u)^\top M M^{-1} G^{-1} M^{-\top} M^\top A_\phi^\top Z \\
&= \boldsymbol{\phi}(u)^\top G^{-1} A_\phi^\top Z,
\end{aligned} \tag{80}$$

which is precisely the predictor constructed in the $\boldsymbol{\phi}$ basis. □

Proposition A.7 shows that changing the basis alone does not change the unregularized polynomial least-squares fit, provided that the design has full column rank and the systems are solved exactly. With rank deficiency, minimum-coefficient-norm solutions can depend on the basis even without regularization. The situation changes when the same identity Tikhonov penalty is imposed in both coefficient systems.

Proposition A.8 (Basis dependence of identity Tikhonov regularization)**.** *Under the assumptions of Proposition A.7, let $\lambda > 0$. Define the identity-Tikhonov predictors*

$$\widehat{z}_{\phi,\lambda}(u) = \boldsymbol{\phi}(u)^\top (A_\phi^\top A_\phi + \lambda I)^{-1} A_\phi^\top Z \tag{81}$$

and

$$\widehat{z}_{\psi,\lambda}(u) = \boldsymbol{\psi}(u)^\top (A_\psi^\top A_\psi + \lambda I)^{-1} A_\psi^\top Z. \tag{82}$$

Expressed in the $\boldsymbol{\phi}$ coordinates, the second predictor is

$$\widehat{z}_{\psi,\lambda}(u) = \boldsymbol{\phi}(u)^\top \left(A_\phi^\top A_\phi + \lambda M^{-\top} M^{-1}\right)^{-1} A_\phi^\top Z. \tag{83}$$

Consequently, after expressing both coefficient estimators in the $\boldsymbol{\phi}$ coordinates, they coincide for every Z if and only if

$$M^{-\top} M^{-1} = I, \tag{84}$$

or equivalently, M is orthogonal. For a general non-orthogonal change of basis, the two regularized forecasters are different despite spanning the same polynomial space.

Proof. Let $C_\psi \in \mathbb{R}^{n\times D}$ be the coefficient matrix in the $\boldsymbol{\psi}$ basis. The identity-Tikhonov objective is

$$\min_{C_\psi} \|A_\psi C_\psi - Z\|_F^2 + \lambda \|C_\psi\|_F^2. \tag{85}$$

Define the equivalent coefficient matrix in the $\boldsymbol{\phi}$ basis by

$$C_\phi = M C_\psi. \tag{86}$$

Using $A_\psi = A_\phi M$, we have

$$A_\psi C_\psi = A_\phi C_\phi, \qquad C_\psi = M^{-1} C_\phi.$$

Thus, Eq. (85) is equivalent to

$$\min_{C_\phi} \|A_\phi C_\phi - Z\|_F^2 + \lambda \|M^{-1} C_\phi\|_F^2. \tag{87}$$

The regularization term can be written as

$$\|M^{-1} C_\phi\|_F^2 = \operatorname{tr}\left(C_\phi^\top M^{-\top} M^{-1} C_\phi\right). \tag{88}$$

The corresponding normal equations are therefore

$$\left(A_\phi^\top A_\phi + \lambda M^{-\top} M^{-1}\right) C_\phi = A_\phi^\top Z, \tag{89}$$

which yields Eq. (83).

By comparison, identity Tikhonov regularization directly in the $\boldsymbol{\phi}$ coordinates gives

$$\left(A_\phi^\top A_\phi + \lambda I\right) C_\phi = A_\phi^\top Z. \tag{90}$$

Since A_ϕ has full column rank, $A_\phi^\top$ is surjective onto $\mathbb{R}^n$. Hence the two coefficient estimators agree for every Z if and only if

$$\left(A_\phi^\top A_\phi + \lambda M^{-\top} M^{-1}\right)^{-1} = \left(A_\phi^\top A_\phi + \lambda I\right)^{-1},$$

which is equivalent to

$$M^{-\top} M^{-1} = I.$$

For an invertible square matrix, this condition is equivalent to M being orthogonal. □

Corollary A.1 (Basis-aware Tikhonov invariance)**.** *Let the Tikhonov matrix in the $\boldsymbol{\phi}$ coordinates be Γ_ϕ. A basis-equivalent regularizer in the $\boldsymbol{\psi}$ coordinates is*

$$\Gamma_\psi = \Gamma_\phi M. \tag{91}$$

More generally, the same quadratic penalty is obtained whenever $\Gamma_\psi^\top \Gamma_\psi = M^\top \Gamma_\phi^\top \Gamma_\phi M$. In particular, if $\Gamma_\phi = I$, choosing $\Gamma_\psi = M$ preserves the penalty; $\Gamma_\psi = I$ does so only when M is orthogonal.

Proof. Since $C_\phi = M C_\psi$,

$$\|\Gamma_\phi C_\phi\|_F^2 = \|\Gamma_\phi M C_\psi\|_F^2.$$

Thus, choosing $\Gamma_\psi = \Gamma_\phi M$ makes both the data-fitting term and the regularization term identical under the change of coordinates. □

Application to Bernstein and shifted-Chebyshev bases. Let

$$\boldsymbol{b}(u) = [b_{0,3}(u), b_{1,3}(u), b_{2,3}(u), b_{3,3}(u)]^\top$$

be the cubic Bernstein basis. Define the first-kind Chebyshev polynomials by $T_0(x) = 1$, $T_1(x) = x$, and $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$ for $k \geq 1$, and let

$$\boldsymbol{c}(u) = [T_0(2u-1), T_1(2u-1), T_2(2u-1), T_3(2u-1)]^\top$$

be the shifted cubic Chebyshev basis. They are related by

$$\boldsymbol{c}(u) = M^\top \boldsymbol{b}(u), \tag{92}$$

where

$$M = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -\frac{1}{3} & -\frac{5}{3} & 5 \\ 1 & \frac{1}{3} & -\frac{5}{3} & -5 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{93}$$

This transformation is not orthogonal; for example, the inner product between its first and third columns is $-4/3$. Therefore,

$$M^{-\top} M^{-1} \neq I.$$

By Proposition A.8, applying the same identity Tikhonov penalty and the same numerical value of λ in the Bernstein and shifted-Chebyshev coefficient systems generally produces different fitted trajectories, different equivalent history weights, and different cache-step predictions.

This distinction does not imply that one basis universally dominates the other. It shows that, once coefficient regularization is introduced, "the choice of basis" and "the choice of regularizer" jointly define the forecasting estimator. Two bases with the same approximation space need not yield the same regularized forecasting operator.

Proposition A.9 (Uniform norm comparison between Bernstein and shifted-Chebyshev bases)**.** *For an integer $d \geq 0$ (so $n = d + 1$), define the degree-d Bernstein basis vector on the normalized coordinate $u \in \mathbb{R}$ as*

$$\mathbf{b}_d(u) = [b_{0,d}(u), \ldots, b_{d,d}(u)]^\top, \qquad b_{j,d}(u) = \binom{d}{j} u^j (1-u)^{d-j}, \tag{94}$$

and define the standard shifted-Chebyshev basis vector as

$$\boldsymbol{\phi}_d(u) = [T_0(2u-1), \ldots, T_d(2u-1)]^\top, \tag{95}$$

where T_k denotes the Chebyshev polynomial of the first kind.

Then, for every $u \in \mathbb{R}$,

$$\boxed{\|\mathbf{b}_d(u)\|_2 \leq \|\boldsymbol{\phi}_d(u)\|_2}. \tag{96}$$

Moreover, the inequality is strict for every $u \in \mathbb{R}$ whenever $d \geq 1$.

Proof. We consider separately the interpolation region $u \in [0,1]$ and the extrapolation region $u \notin [0,1]$.

Case 1: $u \in [0,1]$. The Bernstein basis functions are nonnegative and form a partition of unity:

$$b_{j,d}(u) \geq 0, \qquad \sum_{j=0}^{d} b_{j,d}(u) = 1. \tag{97}$$

Therefore,

$$\|\mathbf{b}_d(u)\|_2 \leq \|\mathbf{b}_d(u)\|_1 = 1. \tag{98}$$

The shifted-Chebyshev vector always contains the constant component

$$T_0(2u-1) = 1, \tag{99}$$

and hence

$$\|\boldsymbol{\phi}_d(u)\|_2 \geq 1. \tag{100}$$

Combining Eqs. (98) and (100) gives

$$\|\mathbf{b}_d(u)\|_2 \leq \|\boldsymbol{\phi}_d(u)\|_2. \tag{101}$$

For $d \geq 1$, the inequality is strict. If $u \in (0,1)$, at least two Bernstein basis entries are positive, so $\|\mathbf{b}_d(u)\|_2 < \|\mathbf{b}_d(u)\|_1 = 1$. At $u \in \{0,1\}$, $\|\mathbf{b}_d(u)\|_2 = 1$, whereas

$$\|\boldsymbol{\phi}_d(u)\|_2 = \sqrt{d+1} > 1.$$

Case 2: $u \notin [0,1]$**.** Let

$$x = |2u - 1| > 1. \tag{102}$$

Using the binomial theorem,

$$\begin{aligned}\|\mathbf{b}_d(u)\|_2 &\le \|\mathbf{b}_d(u)\|_1 \\ &= \sum_{j=0}^{d} \binom{d}{j} |u|^j |1-u|^{d-j} \\ &= \big(|u| + |1-u|\big)^d \\ &= |2u-1|^d = x^d. \end{aligned} \tag{103}$$

The last equality follows because $|u| + |1-u| = |2u-1|$ for $u \notin [0,1]$.

Since the shifted-Chebyshev vector contains its degree-d component,

$$\|\boldsymbol{\phi}_d(u)\|_2 \ge |T_d(2u-1)| = T_d(x), \tag{104}$$

where we used $|T_d(-x)| = |T_d(x)| = T_d(x)$ for $x \ge 1$.

It remains to show that

$$T_d(x) \ge x^d, \qquad x \ge 1. \tag{105}$$

Write $x = \cosh\theta$ for some $\theta \ge 0$. By the identity $T_d(\cosh\theta) = \cosh(d\theta)$,

$$T_d(x) = \cosh(d\theta) = \frac{e^{d\theta} + e^{-d\theta}}{2}. \tag{106}$$

Because $y \mapsto y^d$ is convex on $\mathbb{R}_+$ for integer $d \ge 1$, Jensen's inequality gives

$$\begin{aligned}\frac{e^{d\theta} + e^{-d\theta}}{2} &\ge \left(\frac{e^{\theta} + e^{-\theta}}{2}\right)^d \\ &= \cosh^d\theta = x^d. \end{aligned} \tag{107}$$

Therefore,

$$\|\mathbf{b}_d(u)\|_2 \le x^d \le T_d(x) \le \|\boldsymbol{\phi}_d(u)\|_2. \tag{108}$$

For $d \ge 1$ and $u \notin [0,1]$, all $d+1$ Bernstein entries are nonzero. Hence $\|\mathbf{b}_d(u)\|_2 < \|\mathbf{b}_d(u)\|_1$, making the final inequality strict.

The case $d = 0$ gives equality, since both basis vectors reduce to $[1]$. □

From basis norms to perturbation bounds. For either the standard Bernstein or shifted-Chebyshev basis vector $\varphi(u) \in \mathbb{R}^n$, let row m of A_φ be $\varphi(u_m^{\text{hist}})^\top$. For $\lambda > 0$, define $\omega_{\varphi,\lambda}(u)^\top = \varphi(u)^\top (A_\varphi^\top A_\varphi + \lambda I)^{-1} A_\varphi^\top$. Writing $\widehat{z}_\varphi(u) = \omega_{\varphi,\lambda}(u)^\top Z$, Proposition A.2 gives

$$\|\delta\widehat{z}_\varphi(u)\|_2 \le \|\omega_{\varphi,\lambda}(u)\|_2 \|\delta Z\|_F \le \frac{\|\varphi(u)\|_2}{2\sqrt{\lambda}} \|\delta Z\|_F. \tag{109}$$

For the same degree and numerical identity-Tikhonov coefficient, Proposition A.9 therefore gives

$$\frac{\|\mathbf{b}_d(u)\|_2}{2\sqrt{\lambda}} \le \frac{\|\boldsymbol{\phi}_d(u)\|_2}{2\sqrt{\lambda}}, \qquad u \in \mathbb{R}. \tag{110}$$

This comparison orders the design-matrix-independent sensitivity bounds. The realized weight norms additionally depend on each history design matrix, and prediction accuracy also depends on approximation bias. We measure these quantities directly in Table 9.

A.6 Complementary Roles of Stabilization and Geometric Correction

Proposition A.10 (Large-regularization limit of equivalent weights)**.** *Let $A \in \mathbb{R}^{K\times n}$ be a Bernstein design matrix and let $b(u) \in \mathbb{R}^n$ be the Bernstein basis vector at query coordinate u. Under identity Tikhonov regularization, the equivalent historical weights are*

$$\omega_\lambda(u)^\top = b(u)^\top (A^\top A + \lambda I)^{-1} A^\top. \tag{111}$$

For every fixed A and u,

$$\lim_{\lambda\to\infty} \omega_\lambda(u) = 0. \tag{112}$$

Consequently, increasing λ alone cannot enforce the affine constraint $\mathbf{1}^\top\omega = 1$ or the probability-simplex constraint $\omega \in \Delta$.

Proof. Since $A^\top A$ is positive semidefinite, $\|(A^\top A + \lambda I)^{-1}\|_2 \le 1/\lambda$. Therefore,

$$\|\omega_\lambda(u)\|_2 \le \frac{\|b(u)\|_2 \|A\|_2}{\lambda}, \tag{113}$$

which tends to zero as λ tends to infinity. In particular, $\mathbf{1}^\top\omega_\lambda(u) \to 0$, rather than one. For finite λ, even the scalar design $A = [1]$, $b(u) = [1]$ yields $\omega_\lambda(u) = 1/(1+\lambda) < 1$. Thus coefficient shrinkage alone does not enforce unit sum. Tikhonov stabilization controls fitting sensitivity, while simplex projection in Eq. (18) explicitly enforces the geometry of the conservative historical combination. □

A.6.1 Risk Score and Convex-Hull Fallback

Proof of Proposition 4. The triangle inequality $\|w\|_1 \ge |\mathbf{1}^\top w|$ makes both terms in Eq. (15) nonnegative. If their sum is zero, then $\mathbf{1}^\top w = 1$ and $\|w\|_1 = 1$, implying $w_m \ge 0$ for all m. Thus $w \in \Delta$; conversely, simplex weights make both terms zero. When $\mathbf{1}^\top w = 1$, separating positive and negative entries gives $r_s = \|w\|_1 - 1 = 2\sum_{m=1}^K \max(-w_m, 0)$.

Proof of Proposition 5. Euclidean projection onto the closed convex simplex gives $w_{\text{safe}} \in \Delta$. Hence $z_s^{\text{safe}} = w_{\text{safe}}^\top Z$ is a convex combination of the original cached representations. Equation (20) traces the segment from $\hat{z}_s$ to z_s^{safe} as β_s increases from zero to one. Its endpoint at $\beta_s = 1$ lies in the hull; an intermediate point is not necessarily inside it. For example, with scalar history $Z = [0, 1]^\top$ and raw weights $w = [-1, 2]^\top$, the raw prediction is 2, the projected prediction is 1, and the blend $2 - \beta_s > 1$ lies outside the historical hull $[0, 1]$ whenever $\beta_s < 1$.

A.7 Evaluation Protocol and Run-to-Run Variation

Paired samples and random seeds. The main FLUX evaluation uses 200 DrawBench prompts and the main HunyuanVideo evaluation uses 944 VBench prompts, with five generated samples per prompt from five random seeds. Within each seed, methods share the prompt, initial noise, checkpoint, sampler, sampling grid, and generation settings. Table 7 reports the mean and sample standard deviation of the five seed-level aggregate metrics. Each aggregate is computed over the complete prompt set for that seed. The fidelity ranking is consistent across the seed-level evaluations, and the between-method differences remain larger than the observed seed-level variation.

Main benchmarks and auxiliary evaluations. The main benchmarks use the complete prompt sets above. Additional comparisons and diagnostics include common-subset evaluations, with exactly the same samples, random seeds, and generation settings for all methods within each experiment. We report these as auxiliary evidence separately from the main benchmark table.

Timing protocol. Latency is measured on one NVIDIA H20 96GB GPU with batch size one and matched precision, checkpoints, software environment, and generation settings within each comparison. The timed region covers the complete online sampling loop: full denoiser evaluations, cache maintenance, fitting and forecasting, guardrail computation, the output head, and sampler updates. Model loading, offline metric computation, and preliminary GPU/runtime warm-up iterations are excluded. Algorithmic warm-up full steps remain inside both the NFE count and the timed sampling loop. Each configuration is measured over ten timed runs after runtime warm-up, with CUDA synchronization before and after each run. The reported relative standard deviations lie between 0.1% and 0.4%.

Table 7: Reference fidelity across five paired seeds (mean $\pm$ sample standard deviation). TaylorSeer uses $O = 2$ and $\mathcal{N} = 4$; HiCache uses $O = 2$, $\sigma = 0.5$, and $\mathcal{N} = 4$. BeziCast uses $\mathcal{N} = 4$ on FLUX and $\mathcal{N} = 5$ on HunyuanVideo. These are the less aggressive configurations in Table 1, distinct from the matched-12-NFE comparison.

Model	Method	PSNR ↑	SSIM ↑	LPIPS ↓
FLUX.1-dev	TaylorSeer	19.477 ± 0.45	0.725 ± 0.011	0.144 ± 0.007
	HiCache	17.507 ± 0.42	0.665 ± 0.009	0.206 ± 0.006
	BeziCast	$\mathbf{22.376 \pm 0.41}$	$\mathbf{0.782 \pm 0.012}$	$\mathbf{0.090 \pm 0.006}$
HunyuanVideo	TaylorSeer	20.471 ± 0.46	0.869 ± 0.013	0.290 ± 0.008
	HiCache	20.862 ± 0.47	0.874 ± 0.011	0.275 ± 0.007
	BeziCast	$\mathbf{24.777 \pm 0.50}$	$\mathbf{0.916 \pm 0.011}$	$\mathbf{0.198 \pm 0.008}$

Table 8: End-to-end FLUX.1-dev comparison at 14 full denoiser evaluations. Both methods share the generation and timing settings.

Method	NFE	Latency (s/img) ↓	Speedup ↑	PSNR ↑	SSIM ↑	LPIPS ↓	IR ↑
Spectrum	14	14.30	$2.63\times$	22.77	0.781	0.080	**1.034**
BeziCast	14	**11.58**	$\mathbf{3.25}\times$	**23.38**	**0.796**	**0.075**	1.031

Full-step budget. The matched-budget experiment in Table 2 uses zero-based indices and the exact full-step schedule

$$\mathcal{S}_{\text{full}}^{\text{matched}} = \{0, 1, 2, 3, 4, 10, 16, 22, 28, 34, 40, 46\}. \tag{114}$$

Its five initialization steps and seven subsequent refreshes give 12 full backbone evaluations and 38 cache positions on the 50-step grid. Holding both the count and locations fixed separates forecasting and cache-maintenance costs from the full-denoiser budget. This schedule is a dedicated control experiment; the main benchmark, basis comparison, and guardrail diagnostics retain their separately reported configurations.

A.8 Global and Local Polynomial Forecasters

A.8.1 Matched-Budget Comparison with Spectrum

End-to-end comparison. BeziCast and Spectrum [10] both approximate output-proximal representations as temporal functions, with different history and coordinate choices. We compare the complete methods at 14 full denoiser evaluations using the same FLUX.1-dev checkpoint, DrawBench prompts, initial noise, sampler, resolution, hardware, and timing protocol. Table 8 shows that BeziCast reduces latency from 14.30s to 11.58s and improves all three reference-fidelity metrics, while ImageReward remains similar.

History-dependent overhead. The evaluated Spectrum implementation updates global Chebyshev ridge coefficients from accumulated full-step representations. If the history contains H_s representations of flattened dimension D, forming the degree-d coefficient update involves a high-dimensional product with cost $O((d+1)H_sD)$ and stores $O(H_sD)$ history. History-buffer updates, FP32 conversion, matrix products, and memory allocation contribute to this implementation's online cost. BeziCast instead normalizes time within each recent history window and retains at most $K_{\max} = 6$ full-step representations. Its equivalent-weight form evaluates each forecast in $O(K_{\max}D)$ time with $O(K_{\max}D)$ history storage. This bounded history provides a direct computational advantage as sampling progresses. Table 8 measures the complete pipelines; the local comparison below isolates the regularized temporal parameterization.

A.8.2 Controlled Local Bernstein–Chebyshev Comparison

Matched local predictors. We compare two local identity-Tikhonov predictors with a common output-proximal target, history length $K = 6$, degree $d = 3$, numerical regularization coefficient λ, and fixed 15-NFE schedule. Both use FLUX.1-dev at 1024×1024 on a 50-step grid, with the guardrail disabled. Changing the coefficient basis under identity Tikhonov changes the induced

Table 9: Local basis comparison with common degree, history, regularization coefficient, and 15-NFE schedule; the guardrail is disabled. Output Error is the relative denoising-output error at held-out full steps. Weight norms are measured at actual cache predictions.

Predictor	Output Error ↓	$\|\omega\|_2$ mean/max ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Latency (s/img) ↓
Local Chebyshev	0.247	2.12 / 3.20	24.11	0.817	0.070	12.68
Local Bernstein	**0.211**	**1.60 / 2.87**	**24.70**	**0.830**	**0.056**	12.70

Table 10: Guardrail activation over cache positions. The mean blending strength is computed only over activated positions.

Setting	Cache positions	Activation rate	Mean β_s when activated
FLUX, $\mathcal{N} = 4$	32	31.25%	0.2732
FLUX, $\mathcal{N} = 6$	35	45.71%	0.4344
HunyuanVideo, $\mathcal{N} = 5$	34	35.29%	0.1591
HunyuanVideo, $\mathcal{N} = 6$	35	42.86%	0.1713

regularization geometry, as established in Proposition A.8. Thus, this experiment measures the practical regularized forecasting operators associated with the two parameterizations.

Realized perturbation amplification. For a basis vector $\varphi(u) \in \mathbb{R}^n$, define the fitting operator $\mathcal{S}_\varphi = (A_\varphi^\top A_\varphi + \lambda I)^{-1} A_\varphi^\top \in \mathbb{R}^{n \times K}$, with row m of A_φ equal to $\varphi(u_m^{\text{hist}})^\top$. For fixed $\lambda > 0$, write $\omega_\varphi(u)^\top = \varphi(u)^\top \mathcal{S}_\varphi$, suppressing the λ subscript. A cache perturbation δZ changes the raw prediction $\widehat{z}_\varphi(u) = \omega_\varphi(u)^\top Z$ by

$$\|\delta \widehat{z}_\varphi(u)\|_2 \le \|\omega_\varphi(u)\|_2 \|\delta Z\|_F. \tag{115}$$

The weight norm is the exact operator norm from Frobenius-norm cache perturbations to Euclidean-norm prediction changes. For nonzero weights, equality is attained by $\delta Z = \omega_\varphi(u) v^\top / \|\omega_\varphi(u)\|_2$ for any unit vector $v \in \mathbb{R}^D$. Table 9 shows lower mean and maximum amplification for local Bernstein forecasting, together with lower output error and improved reference fidelity at nearly identical latency.

A.9 Guardrail Activation, Error Association, and Tail Risk

Schedule-dependent activation. Table 10 reports the fraction of cache positions with $\beta_s > 0$; its mean β_s is conditional on activation. For fixed sampling coordinates and fitting parameters, the equivalent weights, risk score, and activation locations are independent of the prompt. The score therefore measures the extrapolation geometry of a query position. Activation becomes more frequent under the larger cache interval, and the conditional mean blending strengths of 0.1591–0.4344 retain a substantial contribution from the raw forecast.

Risk–error association. To measure forecast reliability, we additionally execute the true backbone at each diagnostic cache position using the same accelerated latent state. These diagnostic evaluations are separate from timed inference. We compute raw, pre-guardrail representation and denoising-output errors, average them over paired samples at each position, and measure Spearman correlation across positions. Table 11 shows positive correlations in all four configurations. Positions selected for correction have 1.50–2.18 times larger representation error and 1.63–2.51 times larger output error than inactive positions, linking the coefficient-space risk score to realized forecasting error.

Average and tail fidelity. We compare the raw forecast with risk-adaptive simplex blending using paired prompts and initial noise, identical NFE and schedules, and fixed forecasting hyperparameters. Table 12 reports both average fidelity and the high-error tail through Worst-10% LPIPS and P95 denoising-output error. On HunyuanVideo, blending lowers average LPIPS from 0.146 to 0.129 and Worst-10% LPIPS from 0.362 to 0.250, corresponding to reductions of 11.6% and 30.9%. FLUX also improves in all reported fidelity measures. These paired results identify the guardrail's contribution: selective correction reduces high-error outcomes while retaining the raw local forecaster at low-risk

Table 11: Spearman risk–error correlations and mean error ratios between activated and inactive cache positions. "Repr." and "Output" denote representation and denoising-output error, respectively.

	Spearman correlation		Activated / inactive error	
Setting	Risk–repr.	Risk–output	Repr.	Output
FLUX, $\mathcal{N} = 4$	0.570	0.775	$1.503\times$	$1.630\times$
FLUX, $\mathcal{N} = 6$	0.612	0.862	$2.183\times$	$2.435\times$
HunyuanVideo, $\mathcal{N} = 5$	0.570	0.640	$1.582\times$	$1.965\times$
HunyuanVideo, $\mathcal{N} = 6$	0.642	0.690	$1.659\times$	$2.509\times$

Table 12: Paired raw-versus-guarded diagnostic runs. Worst-10% LPIPS is the mean over the 10% of samples with the highest LPIPS, and P95 denotes the 95th-percentile denoising-output forecasting error. Latency is measured within each diagnostic setting and is separate from the main benchmark configurations.

Setting	Variant	PSNR ↑	LPIPS ↓	Worst-10% LPIPS ↓	P95 error ↓	Latency (s)
FLUX, $\mathcal{N} = 4$	Raw	20.874	0.118	0.251	0.330	12.80
	Risk-adaptive	**21.122**	**0.116**	**0.248**	**0.323**	12.74
HunyuanVideo, $\mathcal{N} = 5$	Raw	26.178	0.146	0.362	0.116	44.64
	Risk-adaptive	**26.959**	**0.129**	**0.250**	**0.112**	44.75

positions. Latency remains close between paired variants, so the small signed timing differences are interpreted as measurement variation rather than a speed gain from correction.

Feature-space and coefficient-space correction. The PCA baseline in Table 18 provides a diagnostic comparison of the correction space. Projection, hull correction, and reconstruction in a retained PCA subspace produce a convex combination of low-rank reconstructions. Simplex projection instead acts on the predictor's existing $K \leq 6$ historical weights and produces an exact convex combination of the original cached representations. This coefficient-space construction preserves all representation directions and avoids selecting a PCA rank. Its alignment with the equivalent-weight risk score explains the more consistent fidelity gains of the simplex variant in that comparison.

A.10 Additional Architectures and Sampler Comparisons

Additional video backbones. We extend the evaluation to CogVideoX1.5-5B [45], LTX-Video-2B [9], and Wan2.1-T2V-14B [39], generating 65, 161, and 81 frames, respectively. Within each model, methods use the same prompts, random seeds, sampler, and generation settings. Table 13 shows that BeziCast achieves the highest speedup and strongest reference fidelity among the compared methods on all three backbones. These results broaden the evidence for local output-proximal forecasting beyond FLUX and HunyuanVideo.

DDIM sampling on DiT-XL/2. We evaluate class-conditional generation on ImageNet at 256×256 with DiT-XL/2 [32] and a 50-step DDIM sampler [37]. At the same 12-full-evaluation budget, BeziCast improves distribution-level FID and sFID over TaylorSeer and HiCache while reducing latency (Table 14). This experiment complements the flow-matching Euler evaluation with a different sampling update and generation task.

A.11 Compatibility with Higher-Order and Variable-Step Samplers

Higher-order updates. BeziCast predicts denoiser representations at the time coordinates requested by the sampler and leaves the solver update unchanged. Let $q \geq 1$ be the number of denoiser evaluations entering an update, indexed by $r = 1, \ldots, q$:

$$\begin{aligned} x_{s+1} &= \Psi_s(x_s, \epsilon_{s,1}, \ldots, \epsilon_{s,q}), \\ \widehat{x}_{s+1} &= \Psi_s(\widehat{x}_s, \widehat{\epsilon}_{s,1}, \ldots, \widehat{\epsilon}_{s,q}). \end{aligned} \tag{116}$$

Here $\epsilon_{s,r}$ and $\widehat{\epsilon}_{s,r}$ are the evaluations used by the full and accelerated solvers, respectively. For a multistep solver, x_s denotes its complete stored state. Define the total evaluation discrepancy

Table 13: Additional text-to-video backbones. Speedup is relative to each model's unaccelerated reference under matched generation settings.

Model	Method	Speedup ↑	PSNR ↑	SSIM ↑	LPIPS ↓
CogVideoX1.5-5B	TeaCache	1.83×	22.29	0.840	0.249
	TaylorSeer	2.91×	23.12	0.872	0.232
	BeziCast	**3.12×**	**24.18**	**0.881**	**0.193**
LTX-Video-2B	TeaCache	1.97×	19.29	0.780	0.381
	TaylorSeer	2.79×	19.86	0.791	0.364
	BeziCast	**3.05×**	**21.41**	**0.823**	**0.315**
Wan2.1-T2V-14B	TeaCache	3.56×	17.96	0.667	0.372
	TaylorSeer	3.89×	18.65	0.689	0.331
	BeziCast	**4.11×**	**21.03**	**0.756**	**0.302**

Table 14: DiT-XL/2 on ImageNet 256×256 with a 50-step DDIM grid and 12 full denoiser evaluations.

Method	NFE	FID ↓	sFID ↓	Latency (s/img) ↓	Speedup ↑
TaylorSeer	12	2.69	5.02	0.26	4.22×
HiCache	12	2.65	4.92	0.26	4.22×
BeziCast	12	**2.53**	**4.84**	**0.23**	**4.78×**

$e_{s,r} = \widehat{\epsilon}_{s,r} - \epsilon_{s,r}$. If Ψ_s has local Lipschitz constants L_s^x and $L_{s,r}^\epsilon$ in the corresponding arguments on the compared paths, then

$$\|\widehat{x}_{s+1} - x_{s+1}\| \le L_s^x \|\widehat{x}_s - x_s\| + \sum_{r=1}^{q} L_{s,r}^\epsilon \|e_{s,r}\|. \tag{117}$$

The discrepancy $e_{s,r}$ includes both same-state forecasting error and drift in stage states; it is not solely a local cache error. Separating these terms requires stage-state sensitivity bounds, as in the single-evaluation derivation of Proposition A.6. Higher-order update coefficients determine how these discrepancies enter the solver. Larger solver steps or stronger stage coupling can increase this sensitivity, while very low-NFE settings reduce the number of full-step representations available for local fitting. A solver-aware schedule should therefore retain exact evaluations during initialization and at solver-critical stages, and invoke forecasting only when sufficient history and a short prediction horizon are available.

Variable-step schedules. BeziCast does not require equally spaced sampling times. With the strictly increasing coordinate τ_s and constant $\varepsilon_\tau > 0$ defined in Section 3.2, historical and unclipped query coordinates are

$$u_m^{\mathrm{hist}} = \frac{\tau_{i_m} - \tau_{i_1}}{\tau_{i_K} - \tau_{i_1} + \varepsilon_\tau}, \qquad \bar{u}_s = \frac{\tau_s - \tau_{i_1}}{\tau_{i_K} - \tau_{i_1} + \varepsilon_\tau}. \tag{118}$$

We then set $u_s = \min(\bar{u}_s, u_{\max})$. Because the denominator includes ε_τ, the latest history coordinate is slightly below one; an unclipped future query exceeds that coordinate but need not exceed one. For multistage solvers with repeated or nonmonotone stage times, this rule requires stage-aware history selection with distinct increasing coordinates, or full evaluations at incompatible stages.

The solver grid and the cache schedule control different error sources: the former controls numerical discretization, whereas the latter controls representation-forecasting error. Consequently, a schedule optimized only for solver truncation error need not be optimal for forecastability. A practical extension is to trigger a full denoiser evaluation when the equivalent-weight risk r_s or the realized amplification factor $\|\omega_\lambda(u_s)\|_2$ exceeds a threshold. Such a rule adapts the full/cache schedule on a given grid; jointly changing the time grid also requires a solver error controller. We leave systematic higher-order and adaptive-sampler evaluation to future work; the present cross-sampler evidence is the DDIM experiment in Table 14.

Comparison with sampler-based acceleration. Table 15 compares forecasting on the DDIM-50 grid with 20- and 10-step DPM-Solver++ [26] on the same DiT-XL/2 model. At similar latency,

Table 15: Sampler-based versus forecasting-based acceleration on DiT-XL/2. This timing comparison uses the reported DDIM-50 reference of 1.09s/image.

Method	Sampling setting	Latency (s/img) ↓	Speedup ↑	FID ↓	sFID ↓
Original	DDIM-50	1.09	1.00×	–	–
DPM-Solver++	20 steps	0.44	2.47×	**2.52**	5.23
DPM-Solver++	10 steps	0.22	4.95×	4.15	6.12
BeziCast	DDIM-50, 12 NFE	0.23	4.74×	2.53	**4.84**

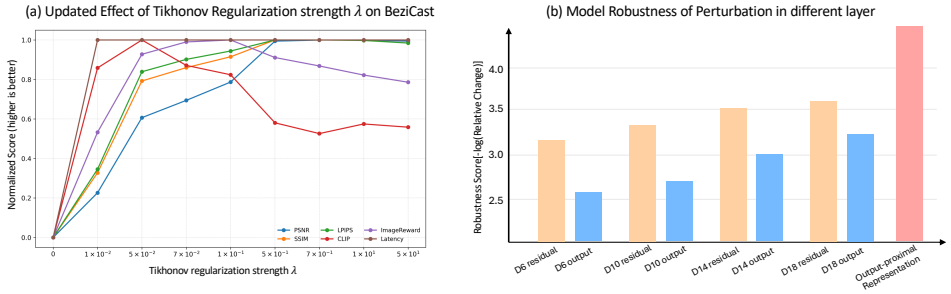


Figure 6: Ablation on Tikhonov regularization and different forecasting targets. (a) Effect of the Tikhonov coefficient λ. (b) Robustness score under equal relative-norm perturbations injected into different FLUX representations; higher values indicate smaller final-output changes.

BeziCast achieves FID/sFID of 2.53/4.84 versus 4.15/6.12 for the 10-step solver. The 20-step solver achieves similar FID (2.52) at 0.44s per image, compared with 0.23s for BeziCast. This comparison evaluates two acceleration routes: reducing solver steps and replacing selected backbone evaluations by representation forecasts. Joint higher-order-solver/cache scheduling is discussed in Appendix A.11.

A.12 Additional Design Ablations

Effect of the ridge coefficient. We further study the effect of the Tikhonov coefficient λ. As shown in Table 16, small values of λ behave similarly to the pseudoinverse estimator and may not sufficiently suppress extrapolation artifacts. Increasing λ reduces control-point magnitude and cache-step error, but overly large λ can over-shrink the trajectory, leading to over-smoothing or loss of fine details. An intermediate regularization strength achieves the best fidelity–stability trade-off as seen in Fig. 6(a).

Number of control representations. The number of control representations n determines the degree and capacity of the Bézier trajectory, while the warm-up length W controls how many initial solver steps are fully evaluated before forecasting begins. We ablate $n \in \{4, 5, 6\}$ under $W = 6$ and $W = 8$, with the Tikhonov coefficient fixed at $\lambda = 5 \times 10^{-1}$. As shown in Table 17, increasing n yields only marginal gains in reference fidelity, while the other quality metrics change little and latency may slightly increase. We therefore use $n = 4$ as the default setting, as it provides a stable cubic trajectory with competitive quality and the lowest or near-lowest latency.

Perturbation injection diagnostic. We further apply the perturbation-injection diagnostic from Appendix A.2 to measure ***how errors at different forecasting targets affect the final denoising output***. Following the local error-propagation analysis in Appendix A.2, intermediate-target errors must traverse the remaining downstream blocks, while output-proximal errors only pass through the final output map. As shown in Fig. 6(b), the final output-proximal representation achieves the highest robustness score among all tested targets, including double-stream block residuals and outputs at layers 6, 10, 14, and 18. Intermediate residuals are generally more robust than block outputs, and later layers are more robust than earlier ones, but all remain less robust than the output-proximal representation. This supports our target choice: BeziCast does not require the final hidden state to be easier to predict, but benefits from its shorter error-propagation path to the denoising output.

Table 16: Effect of regularization strength λ.

λ	PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	IR ↑	Lat. (s) ↓
0	15.425	0.393	0.523	26.590	0.758	15.843
1×10^{-2}	17.148	0.524	0.368	27.331	0.920	15.208
5×10^{-2}	20.043	0.710	0.147	**27.453**	1.040	15.208
1×10^{-1}	21.416	0.759	0.100	27.301	**1.062**	15.208
5×10^{-1}	22.993	**0.793**	**0.075**	27.091	1.035	15.208
1×10^{1}	**23.037**	0.792	0.076	27.086	1.008	15.208

Table 17: Effect of the control point number n.

W	n	PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	IR ↑	Lat. (s) ↓
6	4	20.993	0.750	0.115	27.192	**1.028**	10.109
6	5	21.340	**0.753**	**0.113**	**27.237**	1.018	**10.101**
6	6	**21.401**	0.752	**0.113**	27.201	1.015	**10.101**
8	4	23.287	**0.798**	**0.074**	27.099	1.038	**11.530**
8	5	23.384	**0.798**	0.075	27.125	1.027	11.587
8	6	**23.415**	0.796	0.076	**27.127**	**1.040**	11.595

Feature-space versus coefficient-space guardrails. The PCA comparison isolates the space in which the conservative correction is constructed. Under the $W = 6$ FLUX schedule, we compare no guardrail, a PCA convex-hull guardrail, and the equivalent-weight simplex guardrail. The PCA variant projects cached output-proximal representations into a low-dimensional subspace, applies convex-hull correction there, and reconstructs the result. It serves as a diagnostic feature-space baseline whose geometric constraint applies to the low-rank reconstructions. Simplex projection instead produces a convex combination of the original cached representations without selecting a PCA rank. Table 18 shows that the coefficient-space correction improves PSNR, SSIM, and LPIPS with a small latency increase, supporting the guardrail used in the main method.

A.13 Analysis

Complexity. Let D be the flattened representation dimension, $K_s \le K_{\max}$ the available history length at cache step s, and n the number of Bézier control points. Building the design matrix costs $O(K_s n)$, forming $A^\top A$ costs $O(K_s n^2)$, and factoring the $n \times n$ ridge system costs $O(n^3)$. For each query, solving the factored system and forming the equivalent weights costs $O(n^2 + K_s n)$. Factorizations are reused within a history window. On a fixed time grid and cache schedule, the weights, risk scores, and guardrail coefficients can all be precomputed for each query position; adaptive grids require updating these quantities online. The main high-dimensional operation is the weighted sum $\omega_\lambda(u_s)^\top Z$, which costs $O(K_s D)$. The guardrail adds a simplex projection in K_s dimensions, costing $O(K_s \log K_s)$ with a sorting-based algorithm, and one additional weighted sum. The memory overhead is $O(K_{\max} D)$ for cached output-proximal representations. Algorithm 1 does not store the control representations persistently; prediction uses equivalent historical weights. Our experiments on both FLUX and HunyuanVideo were run on NVIDIA H20 GPU.

Relation to existing predictors. Direct reuse is the constant predictor $\hat{z}_s = z_{i_K}$. Taylor-style forecasting estimates local finite differences and extrapolates them in a power basis. Spectral predictors fit global polynomial bases over the denoising trajectory. BeziCast uses a local function-approximation form: it fits a short-history Bernstein/Bézier trajectory with Tikhonov regularization, evaluates it at cache steps, and writes the result as equivalent weights over historical output-proximal representations. The control-point number n is chosen independently of the cache period.

Limitation. BeziCast is training-free and does not modify the pretrained denoiser or sampler update rule, but its effectiveness relies on the local temporal regularity of output-proximal representations. Abrupt trajectory changes or aggressive cache intervals can still introduce forecasting errors despite Tikhonov stabilization and the risk-adaptive guardrail. Its speed–quality trade-off also depends on the cache schedule and hyperparameters, such as warm-up length, cache period, Bézier degree, and regularization strength, which may require model-specific tuning.

Table 18: Correction-space diagnostic on FLUX.1-dev under the $W = 6$ schedule. Latency is in seconds.

Variant	PSNR ↑	SSIM ↑	LPIPS ↓	CLIP ↑	Rwd. ↑	Lat. ↓
None	20.993	0.750	0.115	27.192	1.028	**10.109**
PCA hull	20.930	0.750	0.115	27.214	1.031	10.150
Simplex	**21.229**	**0.753**	**0.112**	**27.229**	**1.032**	10.137

Our evaluation spans multiple image/video architectures, cache schedules, and both flow-matching Euler and DDIM sampling. Systematic evaluation with higher-order and adaptive samplers, higher resolutions, and longer videos remains open. Finally, BeziCast assumes access to a suitable output-proximal representation and still evaluates the output head at cache steps, which may reduce its applicability or acceleration benefit when such representations are inaccessible or the output head is costly.