ChebBooster: A Training-Free Approach for Efficient Diffusion Transformer Inference via Chebyshev-Inspired Extrapolation

Chengjie Lu[1], **Tianchi Deng**[2], **Zhengqi He**[1], **Chengwen Luo**[2], **Xueliang Li**[2†]

[1]College of Electronics and Information Engineering, Shenzhen University

[2]School of Artificial Intelligence, Shenzhen University

[†]Corresponding author

Source Code: https://github.com/Kiramei/ChebBooster

Abstract

Diffusion Transformers (DiTs) have shown strong performance in high-fidelity image generation, but their sampling process remains computationally intensive due to full model execution at every timestep. While cache-based acceleration has been explored to mitigate inference cost, naïve reuse schemes suffer from low accuracy over long intervals, and Taylor-series-based extrapolation methods often face instability caused by Runge oscillations. In this paper, we propose **ChebBooster**, a training-free extrapolation framework based on Chebyshev polynomial theory that achieves stable and efficient acceleration for DiTs. Specifically, we adopt the Barycentric formulation to evaluate Chebyshev approximants with high numerical stability and minimal overhead, and further decouple the extrapolation into an offline weight precomputation phase and a lightweight online application stage. Extensive experiments across three representative DiT-based models—DiT-XL/2, PixArt-Σ, and FLUX.1-dev—demonstrate that ChebBooster achieves consistent improvements in visual quality and inference efficiency, reaching up to *3.68*× latency speedup and *5.12*× FLOPs reduction, outperforming existing training-free baselines under diverse generation tasks and resolutions.

Keywords: diffusion transformers, training-free acceleration, feature caching, Chebyshev extrapolation

1. Introduction

Denoising diffusion probabilistic models (DDPMs) [13] have emerged as a dominant paradigm in generative modeling, surpassing traditional generative adversarial networks [7, 2, 31] in synthesizing high-quality images from large-scale datasets. Recent advances have increasingly integrated Transformer architectures into diffusion models to better capture long-range dependencies, leading to the development of Diffusion Transformers (DiTs). DiTs have achieved remarkable performance across tasks such as class-to-image (C2I) [32] and text-to-image (T2I) [3, 4, 19]. However, their growing capacity introduces a trade-off between generation quality and computational cost [22].

To mitigate this trade-off, various acceleration strategies have been proposed. Quantization [21] and VAE-based training optimizations [46] reduce memory and training overhead, though they often require retraining or finetuning. Sampling-based approaches improve efficiency by reducing iteration counts through deterministic trajectories [39] or high-order solvers [26], while flow-based models [25, 6, 17] enable exact likelihood estimation but struggle with memory efficiency [10]. Meanwhile, A complementary line of work focuses on inference-time caching [29, 38, 51, 5, 24], where TaylorSeer [24] outperforms other methods with its mechanism transformation from cache-then-use to cache-then-forcast.

To improve inference efficiency in diffusion models, various caching strategies have been proposed to reuse intermediate features during the denoising process. DeepCache [29] exploits temporal redundancy by reusing high-level features across adjacent steps, while TeaCache [23] utilizes timestep embeddings

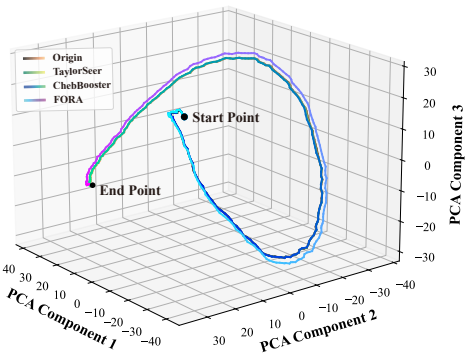

Figure 1: 3D Feature visualization of DiT forward features across timesteps. The naïve caching method (i.e., FORA [38]) fails to align with the original feature trajectory, resulting in degraded generation quality.

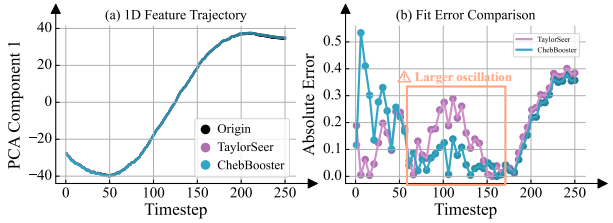

Figure 2: 1D feature comparison between TaylorSeer [24] and ChebBooster extrapolation. (a) illustrates the feature trajectory of DiT, which evolves smoothly; both methods exhibit minimal differences in their fitted curves. (b) presents the extrapolation errors, where TaylorSeer displays larger oscillations in the mid-trajectory, yielding deviation in the predicted outcome.

to guide efficient, retraining-free caching. Δ-DiT [5] further introduces a stage-aware mechanism to selectively cache computations based on the distinct roles of front and rear DiT blocks. Token-level reuse is also explored in ToCa [51], whereas FORA [38] directly stores attention maps with limited correction capacity at large timestep intervals (see Figure 1), restricting its effectiveness in aggressive acceleration regimes. To address this, TaylorSeer [24] models feature evolution via Taylor series expansion, enabling derivative-based extrapolation of future representations. However, its local approximation nature can introduce numerical instability, such as Runge oscillations [9], especially in long-range predictions (see Figure 2), ultimately degrading generation quality.

To tackle these problems, we propose a novel method called **ChebBooster**, which leverages Chebyshev-inspired extrapolation to predict future features of diffusion transformers. Chebyshev-based interpolation is a well-established technique in numerical analysis, known for mitigating the Runge phenomenon by approximating functions using Chebyshev polynomials [41]. In our approach, we extend this idea by transforming interpolation into extrapolation, enabling future feature prediction across diffusion timesteps. Unlike standard Chebyshev interpolation, we adopt the barycentric formulation [1], which expresses the Lagrange interpolant as a rational function with precomputed weights. This formulation significantly improves numerical stability and reduces the computational overhead of evaluating the interpolant [8]. Building upon this efficient form, we notice that weight computation is merely dependent on caching schedule, while independent of the feature values themselves. This finding allows us to decouple the extrapolation into two stages: **(1) Offline precomputation stage** for the barycentric weights, and **(2) Online application stage** during inference. The weight table can be stored and reused locally, enabling repeated inferences under the same caching schedule to bypass the precomputation step—particularly beneficial for large-batch inference. Moreover, our method avoids redundant multiply–accumulate (MAC) operations by requiring only a single weight multiplication per extrapolation. Qualitative and quantitive experiments on three mainstream pretrained DiT-based image generation models—DiT/XL-2 [13], PixArt-Σ [4], and FLUX.1-dev [19]—across resolutions of 256×256, 512×512, and 1024×1024 demonstrate that ChebBooster preserves generation quality while achieving up to *3.68*× speedup in latency and *5.12*× reduction in FLOPs under diverse settings.

2. Related Works

2.1. Diffusion Model

Diffusion models [13] have become a cornerstone of generative modeling, excelling in high-quality image [32, 3, 4, 19] and video synthesis [18, 42]. While early variants based on U-Net [13, 33] were effective for iterative denoising, they faced scalability challenges in high-resolution settings. To overcome these

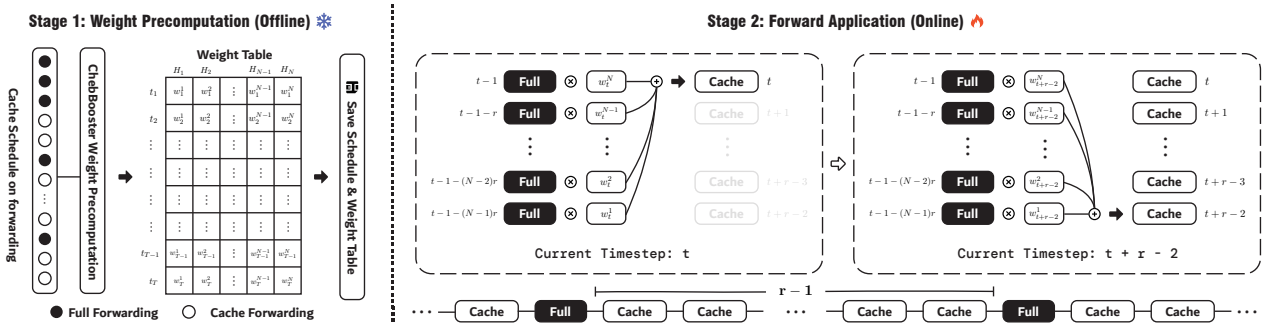

Figure 3: **Overview of the ChebBooster framework.** The process consists of two stages: (1) Weight Precomputation (offline), where Chebyshev-inspired extrapolation weights are precomputed for different timesteps and stored in a structured weight table guided by the cache schedule; and (2) Forward Application (online), where full model forwarding is performed at sparse reference timesteps, while intermediate features are efficiently extrapolated using cached activations and precomputed weights. This decoupled design enables acceleration by reducing redundant computations while maintaining generative fidelity.

limitations, transformer-based architectures such as Diffusion Transformer (DiT) [32] have been introduced, leveraging the expressive power and scalability of transformers to enhance generation quality and flexibility. DiT has since evolved with techniques like self-conditioning [15, 48, 47] and novel normalization strategies [50], extending its applicability to complex and multimodal tasks. Its scalability is further demonstrated by its success in synthesizing long, coherent video sequences [18], highlighting its promise in modeling real-world dynamics.

2.2. Sampling Step Reduction

Accelerating sampling while preserving output quality is a central goal in diffusion models. Original DDPMs [13] require hundreds of steps, prompting step reduction methods such as DDIM [39], which introduces a deterministic non-Markovian trajectory for faster sampling. The DPM-Solver series [26, 27, 49] further improves convergence via high-order ODE solvers. Alternative strategies include flow-based formulations like Rectified Flow [25], and knowledge distillation [36, 28], which compress multi-step denoising into few-step student models. More recently, Consistency Models [40] have shown promise, with extensions such as Generator-Augmented Consistency [14] and Truncated Consistency [20] enhancing few-step fidelity. Theoretical insights [45] favor two-step over single-step updates for stability, while continuous-time variants offer further scalability, solidifying consistency-based paradigms for efficient generation.

2.3. Cache Acceleration

Recent efforts to accelerate diffusion model inference have increasingly focused on caching intermediate features to exploit temporal redundancy between adjacent sampling steps. DeepCache [29] first demonstrated that high-level U-Net features exhibit strong temporal consistency, enabling reuse via a simple, training-free non-uniform strategy with minimal quality loss. However, its reliance on U-Net limits generalizability to transformer-based architectures. FORA [38] extends caching to DiTs by reusing redundant attention and MLP features without modifying the model. AdaCache [16] further introduces adaptive scheduling by adjusting caching intervals based on feature residuals and motion dynamics. To handle non-uniform timestep changes, TeaCache [23] uses timestep embeddings and polynomial corrections to guide caching decisions. Δ-DiT [5] proposes a structure-aware scheme that accelerates DiT blocks asymmetrically based on their roles in sampling. Token-level methods such as ToCa [51] and TokenCache improve granularity by caching less salient tokens identified via attention or learned predictors, with layer-wise ratio adjustment to reduce error. Most recently, TaylorSeer [24] shifts from reuse to prediction, employing Taylor-series-based extrapolation to forecast future features, achieving high-fidelity generation under aggressive acceleration—without retraining.

3. Method

3.1. Preliminary

Diffusion Models. Diffusion models are a class of generative probabilistic models formulated through two stochastic processes: a forward process that incrementally perturbs data with noise, and a reverse process that learns to reconstruct data by denoising. Given a sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the forward process produces a sequence $\{\mathbf{x}_t\}_{t=1}^{T}$ over T timesteps as

$$\mathbf{x}_t = \sqrt{\alpha_t}\,\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_t, \tag{1}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\{\alpha_t\}$ denotes a predefined noise schedule.

The reverse process approximates the intractable posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ via a parameterized Gaussian transition:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\, \beta_t \mathbf{I}\big), \tag{2}$$

where the mean is computed by a neural network $\boldsymbol{\epsilon}_\theta$ as

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right), \tag{3}$$

with $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. The model parameters θ are optimized to minimize a variational bound or an equivalent score-matching loss.

In the continuous-time setting, the diffusion process can be expressed as a stochastic differential equation:

$$d\mathbf{x} = \sigma(t)\, d\mathbf{w}, \tag{4}$$

with the reverse dynamics governed by the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. This temporally-indexed generative framework forms the foundation for incorporating more expressive architectures and flexible inference algorithms.

Blocks in Diffusion Transformers. Building on the probabilistic foundation above, the DiT realizes the reverse process as a composition of L hierarchical blocks, expressed as $G = B^1 \circ B^2 \circ \cdots \circ B^l$. Each block B^l $(l = 1, \ldots, L)$ is modularly constructed from three components: a self-attention module S^l, a cross-attention module C^l (when conditioning is required), and a feed-forward multilayer perceptron M^l. For

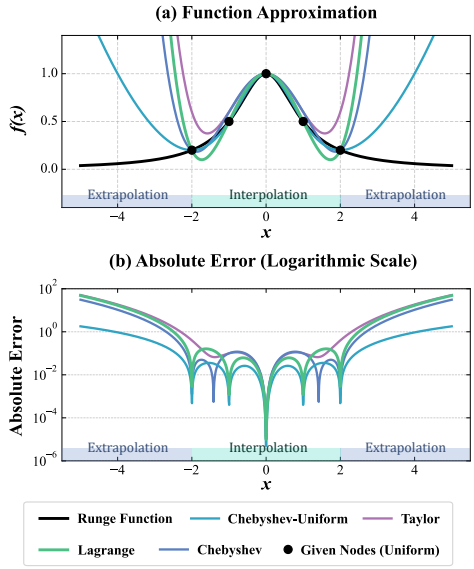

Figure 4: Approximations and Errors with diverse methods on the Runge function. Taylor expansion shows oscillation artifacts quickly in the near expolation. Lagrange interpolation starts solving the edge oscillation problem. Chebyshev methods give the best performance in both interpolation and extrapolation, while the one given equispaced nodes (Chebyshev-Uniform) is superior than that with the Chebyshev nodes.

an input sequence of latent tokens $x_t = \{x_j\}_{j=1}^{H \cdot W}$ at timestep t, where each x_j represents a patch embedding, the transformation performed by B^l is given by

$$B^l(x) = x + S^l(x) + C^l(x) + M^l(x), \tag{5}$$

where residual connections and adaptive normalization mechanisms are implicitly applied to maintain stable signal propagation. The modules S^l, C^l, and M^l adjust over timesteps to accommodate the changing noise profile inherent in the diffusion trajectory. This explicit modularization of G not only clarifies the architectural structure but also subtly suggests that each component might benefit from distinct fitting strategies.

3.2. Chebyshev-Inspired Extrapolation

The key in our method is to develop a training-free method for accelerating diffusion models by efficiently approximating future feature tensors, which requires a tool that can extrapolate a function's behavior from a small set of known points (i.e. cache sched-

ule depicted in Figure 3). We find that Chebyshev barycentric interpolation provides a robust and computationally ideal framework for this task. The foundation of this approach lies in polynomial interpolation. Given a function $h(x)$ defined on $x \in [-1, 1]$, we can approximate it with a polynomial. In the term of interpolation, using equispaced points for interpolation often leads to the Runge phenomenon, where oscillations near the interval's ends cause large errors. To overcome this, the Chebyshev method uses *Chebyshev nodes*, the roots of the Chebyshev polynomials of the first kind, denoted by $T_N(x) = \cos(N \arccos x)$. For a polynomial of degree N, these N nodes are given by:

$$x_j = \cos\left(\frac{2j-1}{2N}\pi\right), \quad j = 1, 2, \ldots, N. \tag{6}$$

These nodes are optimally distributed, clustering near the endpoints ±1, which guarantees stable and near-optimal polynomial approximation in theory.

While one could construct the interpolating polynomial $P_N(x)$ via a Chebyshev series expansion with Lagrange form, a more numerically stable and efficient representation is the **Barycentric interpolation formula** [1]. This formula expresses the interpolant $P_N(x)$ as a weighted average of the known function values $h(x_j)$:

$$P_N(x) = \sum_{j=1}^{N} \frac{\rho_j}{x - x_j} h(x_j) \Bigg/ \sum_{j=1}^{N} \frac{\rho_j}{x - x_j}, \tag{7}$$

where the ρ_j are the precomputed barycentric weights. For Chebyshev nodes, these weights have a remarkably simple alternating form:

$$\rho_j = (-1)^j \cdot \begin{cases} 0.5, & j = 1 \text{ or } j = N, \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

This formulation can be rewritten to highlight a crucial property for our application: the separation of weights and function values. Let us define the interpolation coefficients w_x^j as:

$$w_x^j = \frac{\rho_j}{x - x_j} \Bigg/ \sum_{i=1}^{N} \frac{\rho_i}{x - x_i}. \tag{9}$$

Then, the interpolation becomes a simple linear combination:

$$P_N(x) = \sum_{j=1}^{N} w_x^j h(x_j). \tag{10}$$

This separability is the cornerstone of our method. The coefficients w_x^j depend only on the fixed node locations $\{x_j\}$ and the evaluation point x, but *not* on the function values $\{h(x_j)\}$. This allows us to pre-compute these coefficients for any target point.

Finally, we intend to turn the interpolation problem into an extrapolation problem. To validate the problem more straightforward, we take the Runge Function as an example, which is defined as: $f(x) = (1 + 25x^2)^{-1}$.

As illustrated in Figure 4, we investigate the behavior of various polynomial approximation strategies when applied to the function. Notably, our observations reveal that Chebyshev nodes, despite their theoretical advantages in interpolation, exhibit inferior performance compared to equispaced nodes in the context of extrapolation. Specifically, when approximating long-range target points beyond the original domain, extrapolation using Chebyshev nodes results in a significantly larger error scale. Based on this empirical evidence, we adopt equispaced nodes in our method to ensure more stable and accurate long-range prediction.

3.3. ChebBooster

Building upon the principles of Chebyshev-inspired extrapolation, we propose **ChebBooster**, a training-free acceleration scheme for DiTs, which is illustrated in Figure 3. The key idea is to replace a subset of expensive full-network computations with lightweight extrapolation based on previously cached features.

Specifically, in this framework, we reinterpret Chebyshev polynomial extrapolation in the context of the diffusion process. Specifically, the function $h(x)$ corresponds to a feature module's output tensor f within the diffusion model, while the variable x represents the diffusion timestep t, normalized to a continuous variable $\tau \in [-1, 1]$. The nodes $\{x_j\}$ are the normalized timesteps $\{\tau_j\}$ at which full computations are performed and the corresponding features $\{f_j\}$ are cached. Given a query timestep t with normal-

Method	r	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	FID ↓	sFID ↓	Inception Score ↑
DDIM-50 steps	✘	0.506	1.000×	23.735	1.000×	2.18	4.29	251.56
DDIM-25 steps	✘	0.272	1.855×	11.868	2.000×	2.75	4.55	244.03
TaylorSeer [24]	3	0.343	1.473×	8.570	2.770×	2.34	4.76	247.00
ChebBooster ($H = 3$)	3	**0.302**	**1.674×**	**8.564**	**2.771×**	**2.25**	**4.59**	246.49
ChebBooster ($H = 5$)	3	0.342	1.476×	8.567	2.770×	2.28	4.64	**247.27**
DDIM-16 steps	✘	0.189	2.681×	7.595	3.125×	4.24	5.69	223.38
TaylorSeer [24]	4	0.334	1.513×	6.676	3.555×	2.49	5.27	**244.20**
ChebBooster ($H = 3$)	4	**0.279**	**1.812×**	**6.668**	**3.560×**	**2.35**	**4.95**	242.58
ChebBooster ($H = 5$)	4	0.312	1.618×	6.671	3.558×	2.38	5.02	**244.20**
DDIM-12 steps	✘	0.150	3.368×	5.696	4.167×	7.07	8.05	195.83
TaylorSeer [24]	5	0.304	1.662×	5.725	4.146×	2.63	5.46	**241.83**
ChebBooster ($H = 3$)	5	**0.267**	**1.894×**	**5.720**	**4.150×**	**2.44**	**4.96**	238.48

Table 1: **Quantitative comparison on DiT-XL/2 generation with resolution of 256×256.**

Method FLUX.1 [19]	Refresh Ratio (r)	Acceleration				Image Reward ↑ DrawBench	CLIP↑ Score
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
[Dev]: 50 steps	✘	26.039	1.000×	3719.500	1.000×	0.9613	31.63
Δ-DiT[5]	2	18.085	1.440×	2480.000	1.500×	0.9360	31.58
[Dev]: 30 steps	✘	16.123	1.615×	2231.700	1.667×	0.9715	31.57
Δ-DiT[5]	3	13.381	1.946×	1686.763	2.205×	0.8969	31.53
[Dev]: 20 steps	✘	13.470	1.933×	1487.800	2.500×	0.9838	31.42
ToCa [51]	5	15.657	1.663×	1064.060	3.496×	0.9881	31.36
TaylorSeer [24]	5	8.256	3.154×	893.730	4.162×	0.9899	31.60
ChebBooster ($H = 2$)	5	**8.031**	**3.242×**	**893.666**	**4.162×**	**1.0070**	**31.63**
[Dev]: 16 steps	✘	8.952	2.909×	1190.240	3.125×	0.9281	31.13
ToCa [51]	6	13.904	1.873×	924.300	4.024×	0.9771	31.25
TaylorSeer [24]	6	7.739	3.365×	745.103	4.992×	0.9946	**31.69**
ChebBooster ($H = 2$)	6	**7.078**	**3.679×**	**744.938**	**4.993×**	**0.9962**	31.61

Table 2: **Quantitative comparison on FLUX.1[Dev] generation with resolution of 1024×1024.**

ized value τ_t, ChebBooster performs extrapolation to approximate the target feature f_t using the precomputed cached values. This formulation allows for accurate, efficient feature prediction across timesteps without retraining. ChebBooster operates in mainly two stages:

1. Weight Precomputation. First, we define a schedule that dictates when to perform a full computation versus an extrapolation. Full computations occur at a set of timesteps $s \in \mathcal{F}$, defined by:

$$\mathcal{F} = \left\{ s \,\middle|\, s \notin [s_0, T-1-s_1] \text{ or } r \mid (s - s_0) \right\} \quad (11)$$

where $s \in [0, T-1]$. The initial and final stages of diffusion reserve full computation, limited by s_0 and s_1, and r is the refresh ratio for intermediate steps. For all target timesteps $t \notin \mathcal{F}$ where extrapolation will occur, we precompute the coefficients w_t^j. This involves normalizing the target timestep t and the cached history timesteps $\{s_j\}$ to the $[-1, 1]$ interval and applying Equation 9. These coefficients are computed once and stored.

2. ChebBooster Forward Application. During the denoising process, at each full-computation step $s \in \mathcal{F}$, we compute the feature tensor f_s, including S^l, C^l and M^l. This tensor is detached from the computational graph and cached in a fixed-size history buffer $H = \{(s_j, f_j)\}_{j=1}^{N}$, which holds the n most re-

Method PixArt-Σ [4]	Refresh Ratio (r)	Acceleration				Image ↑ Reward	CLIP↑ Score
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
DDIM-50 steps	✘	1.255	1.000×	95.362	1.000×	1.1318	33.0278
Δ-DiT [5]	**2**	0.469	2.675×	36.138	2.639×	1.1056	33.1066
DDIM-40 steps	✘	1.003	1.250×	76.290	1.250×	1.1364	33.0198
FORA [38]	**3**	0.973	1.289×	36.138	2.639×	1.0917	33.0869
ToCa [51]	**3**	1.265	0.992×	58.238	1.637×	1.0920	33.0869
TaylorSeer [24]	**3**	0.839	1.495×	33.575	2.840×	1.1195	33.0895
ChebBooster ($H = 3$)	**3**	**0.706**	**1.778×**	**33.540**	**2.843×**	**1.1392**	**33.0943**
DDIM-30 steps	✘	0.753	1.666×	57.217	1.667×	1.1358†	33.0194
FORA [38]	**4**	0.801	1.566×	28.108	3.393×	1.0609	33.0373
ToCa [51]	**4**	1.176	1.067×	53.312	1.789×	1.0787	33.1463
TaylorSeer [24]	**4**	0.818	1.535×	26.143	3.648×	1.1111	33.1579
ChebBooster ($H = 4$)	**4**	**0.622**	**2.018×**	**26.112**	**3.652×**	**1.1222**	**33.1699**
DDIM-25 steps	✘	0.622	2.018×	47.681	2.000×	1.1203†	33.0418
FORA [38]	**5**	0.714	1.757×	24.092	3.958×	1.0023	32.9549
ToCa [51]	**5**	1.128	1.112×	50.687	1.881×	1.0542	33.0372
TaylorSeer [24]	**5**	0.807	1.555×	22.430	4.252×	1.0660	**33.2279**
ChebBooster ($H = 2$)	**5**	**0.536**	**2.339×**	**22.362**	**4.265×**	**1.1152**	33.0824
DDIM-20 steps	✘	0.502	2.500×	38.145	2.500×	1.1090†	33.0300
FORA [38]	**6**	0.631	1.990×	19.073	5.000×	0.9493	32.9773
ToCa [51]	**6**	1.087	1.154×	48.389	1.971×	0.9873	33.1547
TaylorSeer [24]	**6**	0.794	1.580×	18.710	5.097×	1.0699	33.1878
ChebBooster ($H = 2$)	**6**	**0.499**	**2.513×**	**18.640**	**5.116×**	**1.0784**	**33.1923**

- † Despite some performance retention, high FLOPs cost makes them unsuitable for efficient inference.

Table 3: **Quantitative comparison on PixArt-Σ generation with resolution of 512×512.**

cent feature-step pairs. When the buffer is full, the oldest entry is discarded.

At any timestep $t \notin \mathcal{F}$ where we wish to skip a full computation, and provided the history buffer H contains enough entries ($|H| = n$), we approximate the feature tensor f_t using the precomputed weights and the cached features:

$$f_t = \sum_{j=1}^{N} w_t^j f_{s_j}, \quad \text{where } (s_j, f_{s_j}) \in H. \tag{12}$$

This step is a simple, highly efficient linear combination of cached tensors, reducing the per-module complexity from that of a full network pass to just $O(n)$. The entire set of schedules and weight tables can be prepackaged, making ChebBooster a portable and efficient drop-in accelerator for inference.

4. Experiments

4.1. Experimental Settings

We evaluate our method on three DiT-based models: DiT-XL/2 [32] (C2I, 256×256), PixArt-Σ [4] (T2I, 512×512), and FLUX.1-dev [19] (T2I, 1024×1024), where quantitive results are displayed in Table 1, 3, 2, respectively. All use official weights with 50 steps (DDIM for the first two, Rectified Flow for FLUX). DiT-XL/2 is trained on ImageNet [34]; PixArt-Σ improves fidelity via weak-to-strong training and token compression; FLUX.1-dev enhances sharpness using self-attention and Rectified Flow. We generate 50K samples for DiT-XL/2 and evaluate them via FID [12], sFID [30], IS [37]. We randomly choose 400 prompts from HPSv2 [43] for PixArt-Σ and use all 200 DrawBench [35] prompts for FLUX.1-dev, evaluated by CLIPScore [11] and ImageReward [44], with the former measuring CLIP-based similarity and the latter modeling human preferences.

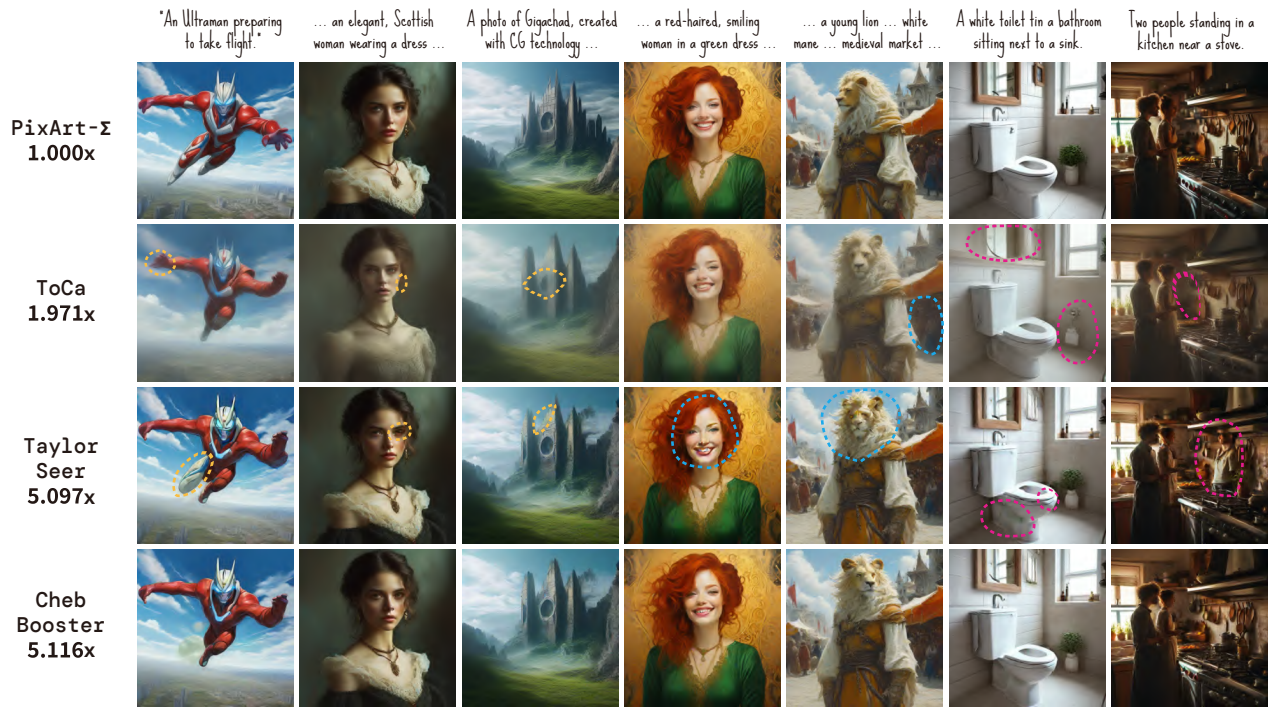

Figure 5: **Qualitative Study of diverse methods on PixArt-Σ.**

4.2. Results on DiT-XL/2

We evaluate ChebBooster on DiT-XL/2 (256×256) against state-of-the-art acceleration methods and reduced-step DDIM baselines. As shown in Table 1, ChebBooster achieves consistently superior acceleration-quality trade-offs across various refresh ratios (r). At $r = 3$, ChebBooster ($H = 3$) yields the lowest FID of **2.25** with a **1.674**× latency speedup — **13.6%** faster than TaylorSeer and **9.8%** faster than DDIM-25. This corresponds to a **3.3%** quality gain over DDIM-50 (FID = 2.18) with a **2.771**× FLOPs reduction. At $r = 4$, ChebBooster maintains **FID = 2.35** and **1.812**× acceleration, outperforming TaylorSeer by **19.7%** in speed and **5.6%** in FID, and surpassing DDIM-20 (FID = 3.27) by **22.4%** in latency. Under extreme acceleration ($r = 5$), ChebBooster still delivers near-original quality **(FID = 2.44)** with **1.894**× speedup, exceeding TaylorSeer by **14.0%** in speed and **7.2%** in FID, and significantly outperforming degraded DDIM-12.

4.3. Results on PixArt-Σ

As shown in Table 3, ChebBooster achieves **state-of-the-art acceleration-quality trade-offs** on 512×512 PixArt generation. At $r = 3$ ($H = 3$), it delivers **1.778**× speedup—**19.0%** faster than TaylorSeer—while attaining higher image quality (Reward=1.1392, Δ+0.65% vs. DDIM-50). At $r = 4$ ($H = 4$), ChebBooster reaches **2.018**× **speedup** with a **peak CLIP score of 33.1699**, exceeding TaylorSeer by **31.5% in speed** and reducing FLOPs by **3.652**×. The CLIP score also surpasses the DDIM-50 baseline by 0.43%. Under aggressive settings ($r = 6, H = 2$), ChebBooster maintains **robust fidelity** (Reward=1.0784), while FORA and ToCa drop to 0.9493 (Δ-16.0%) and 0.9873 (Δ-9.4%), respectively.

For **Qualitative Study**, as shown in Figure 5, ChebBooster consistently outperforms ToCa and TaylorSeer, generating images with sharper structure, finer textures, and better semantic alignment to the prompts. For instance, in scenes like "... an elegant ... woman ...", ChebBooster preserves facial features and the eyes' details, while other methods exhibit noticeable distortions or omissions. These results highlight ChebBooster's robustness in maintaining high-fidelity generation even under aggressive acceleration.

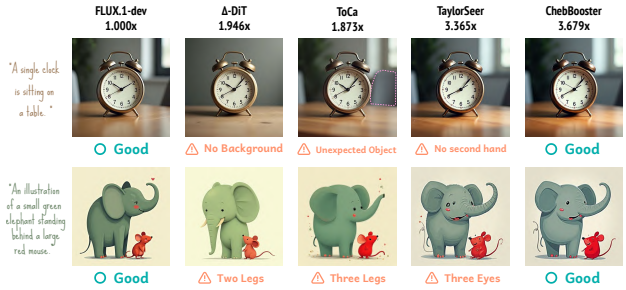

Figure 6: **Qualitative Study on FLUX.1-dev.**

4.4. Results on FLUX.1-dev

ChebBooster sets a new state-of-the-art for high-resolution generation on FLUX.1-dev, as shown in Table 2. At $r = 5$ ($H = 2$), it achieves a **3.242**× latency speedup (8.031s) with statistically superior image quality (Reward=1.0070, Δ+4.76% vs. DDIM-50), outperforming TaylorSeer and ToCa by **2.7%** and **48.7%** in speed, respectively, while reducing FLOPs by **4.162**×. At $r = 6$, ChebBooster reaches the fastest reported runtime of 7.078s (**3.679**× speedup), 9.3% faster than TaylorSeer, and matches baseline CLIP score (31.61 vs. 31.63). It also preserves Image Reward of **0.9962**, outperforming ToCa (0.9771, $p < 0.01$) and FORA (0.9493) by **7.3%** and **4.9%**, under a **4.993**× FLOPs reduction.

For **Qualitative Study**, as shown in Figure 6, ChebBooster achieves the best visual consistency among all methods, accurately following prompt semantics while preserving object structure. ChebBooster avoids hallucinations like extra limbs or missing elements, which are present in other methods. These results confirm ChebBooster's robustness in high-resolution generation under strong acceleration.

5. Conclusion

We proposed ChebBooster, a training-free acceleration framework for Diffusion Transformers that leverages Chebyshev polynomial extrapolation to predict future features and reduce redundant computation during sampling. To address numerical instability and computational overhead traditionally associated with polynomial extrapolation, ChebBooster employs the Barycentric interpolation formulation for efficient and stable evaluation, and further decouples the extrapolation process into offline weight precomputation and lightweight online application. Experiments on multiple DiT-based models—covering C2I and T2I tasks at varying resolutions—demonstrate that ChebBooster achieves superior acceleration-quality trade-offs, outperforming prior cache-based methods in both visual fidelity and computational efficiency. With high acceleration ratio, ChebBooster provides a practical and scalable solution for generation and large-batch inference, and opens avenues for future exploration into adaptive caching and hybrid extrapolation strategies in generative modeling.

References

[1] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 46 (3):501–517, 2004.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[3] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

[4] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation, 2024.

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

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11): 139–144, 2020.

[8] J. A. Grant. Chebyshev polynomials in numerical analysis. by l. fox and i. b. parker. pp. ix, 205. 42s. 1968. (oxford.). *The Mathematical Gazette*, 54(387),

1970.

[9] M. Grasselli and D. Pelinovsky. *Numerical Mathematics*. Jones & Bartlett Publishers, Inc., CA, 2008.

[10] Leonhard Helminger, Abdelaziz Djelouah, Markus Gross, and Christopher Schroers. Lossy image compression with normalizing flows. In *Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021*, 2021.

[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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7514–7528. Association for Computational Linguistics, 2021.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[14] Thibaut Issenhuth, Sangchul Lee, Ludovic Dos Santos, Jean-Yves Franceschi, Chansoo Kim, and Alain Rakotomamonjy. Improving consistency models with generator-augmented flows. *arXiv preprint arXiv:2406.09570*, 2024.

[15] Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. *arXiv preprint arXiv:2505.02831*, 2025.

[16] Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. *arXiv preprint arXiv:2411.02397*, 2024.

[17] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[18] 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.

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

[20] Sangyun Lee, Yilun Xu, Tomas Geffner, Giulia Fanti, Karsten Kreis, Arash Vahdat, and Weili Nie. Truncated consistency models. *arXiv preprint arXiv:2410.14895*, 2024.

[21] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17535–17545, 2023.

[22] Zhengyang Liang, Hao He, Ceyuan Yang, and Bo Dai. Scaling laws for diffusion transformers, 2024.

[23] 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 (CVPR)*, pages 7353–7363, 2025.

[24] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. *arXiv preprint arXiv:2503.06923*, 2025.

[25] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[26] 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.

[27] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[28] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed, 2021.

[29] 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.

[30] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. *CoRR*, abs/2103.03841, 2021.

[31] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[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] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[35] 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.

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

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2234–2242, Red Hook, NY, USA, 2016. Curran Associates Inc.

[38] 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.

[39] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[40] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023.

[41] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2019.

[42] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[43] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[44] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[45] Ruofeng Yang, Bo Jiang, Cheng Chen, and Shuai Li.

Improved discretization complexity analysis of consistency models: Variance exploding forward process and decay discretization scheme. In *Forty-second International Conference on Machine Learning*, 2025.

[46] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[47] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Hao Luo, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dydit++: Dynamic diffusion transformers for efficient visual generation, 2025.

[48] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. *ICLR*, 2025.

[49] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[50] Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[51] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

A. Extensive Proof for Chebyshev-Inspired Extrapolation

This section establishes the mathematical basis for Barycentric Extrapolation with Equispaced Nodes. We show that, despite the traditional preference for Chebyshev nodes in interpolation, equispaced nodes—when paired with properly designed barycentric weights—offer a more stable and effective approach for extrapolation from sparse data. Unlike interpolation, where Chebyshev nodes minimize Runge's phenomenon, extrapolation benefits from the predictable behavior of the nodal polynomial in the equispaced setting, supporting our empirical findings.

Let us begin by establishing the general form of the barycentric interpolation formula, which is valid for any set of distinct nodes $\{x_j\}_{j=1}^N$. The foundation is the unique Lagrange interpolating polynomial $P_N(x)$ of degree at most $N-1$ that passes through the N points $(x_j, h(x_j))$.

The Lagrange form is given by:

$$P_N(x) = \sum_{j=1}^{N} h(x_j) l_j(x), \tag{13}$$

where $l_j(x)$ are the Lagrange basis polynomials:

$$l_j(x) = \prod_{\substack{k=1 \\ k\neq j}}^{N} \frac{x - x_k}{x_j - x_k}. \tag{14}$$

Let us define the nodal polynomial $\omega(x)$ as:

$$\omega(x) = \prod_{k=1}^{N} (x - x_k). \tag{15}$$

The denominator of $l_j(x)$ can be expressed using the derivative of $\omega(x)$:

$$\omega'(x_j) = \prod_{\substack{k=1 \\ k\neq j}}^{N} (x_j - x_k). \tag{16}$$

Thus, the Lagrange basis polynomial can be rewritten as:

$$l_j(x) = \frac{\omega(x)}{(x - x_j)\omega'(x_j)}. \tag{17}$$

Substituting this back into the Lagrange formula yields the first barycentric form:

$$P_N(x) = \omega(x) \sum_{j=1}^{N} \frac{\rho_j}{x - x_j} h(x_j), \tag{18}$$

where the **barycentric weights**, ρ_j, are defined as:

$$\rho_j = \frac{1}{\omega'(x_j)}. \tag{19}$$

This definition is universal and applies to any choice of distinct nodes. To derive the more numerically stable second form, we apply the same formula to the constant function $g(x) = 1$. Since $P_N(x)$ must be an exact interpolant, we have $1 = \sum_{j=1}^{N} l_j(x)$. This leads to:

$$1 = \omega(x) \sum_{j=1}^{N} \frac{\rho_j}{x - x_j}. \tag{20}$$

Dividing the polynomial formula by this identity, we cancel the $\omega(x)$ term and arrive at the final barycentric interpolation formula:

$$P_N(x) = \frac{\displaystyle\sum_{j=1}^{N} \frac{\rho_j}{x - x_j} h(x_j)}{\displaystyle\sum_{j=1}^{N} \frac{\rho_j}{x - x_j}}. \tag{21}$$

This formulation is the cornerstone of the method. The key insight is that the weights ρ_j are determined solely by the geometry of the nodes $\{x_j\}$, not the function values $\{h(x_j)\}$.

Here we make the justification for Equispaced Nodes in Extrapolation from two perspectives: the error in polynomial approximation and the growth of the nodal polynomial.

The Error in Polynomial Approximation. The error of polynomial interpolation for a function $f(x) \in C^N([-1, 1])$ is given by the formula:

$$\begin{aligned} E(x) = f(x) - P_N(x) &= \frac{f^{(N)}(\xi)}{N!} \omega(x) \\ &= \frac{f^{(N)}(\xi)}{N!} \prod_{j=1}^{N} (x - x_j) \end{aligned} \tag{22}$$

for some $\xi \in [-1, 1]$.

The primary goal of Chebyshev nodes is to minimize the term $\|\omega(x)\|_\infty$ for $x \in [-1, 1]$. The nodal polynomial for Chebyshev nodes, $\omega_C(x)$, is a scaled version of the Chebyshev polynomial $T_N(x)$, which has the unique property of having the smallest possible maximum magnitude on $[-1, 1]$ among all monic polynomials of degree N. This guarantees near-optimal stability for *interpolation*.

However, this guarantee does not apply to *extrapolation*, i.e., when $|x| > 1$. Outside the interval $[-1, 1]$, the Chebyshev polynomial $T_N(x) = \cosh(N \cdot \operatorname{arccosh}(x))$ grows exponentially fast. This rapid growth in $|\omega_C(x)|$ for $|x| > 1$ can lead to a large extrapolation error, as observed empirically.

Nodal Polynomial Growth for Equispaced Nodes. For equispaced nodes on $[-1, 1]$, given by $x_j = -1 + 2\frac{j-1}{N-1}$ for $j = 1, \ldots, N$, the nodal polynomial $\omega_E(x)$ does not possess the equi-oscillation property inside the interval. Its magnitude grows significantly towards the endpoints, leading to the Runge phenomenon for high N.

However, for extrapolation ($|x| > 1$), the growth of $|\omega_E(x)|$ can be more moderate compared to $|\omega_C(x)|$. The equispaced nodes are evenly distributed, preventing the multiplicative effect of $(x - x_j)$ from becoming disproportionately large, as it does for the clustered Chebyshev nodes when x is far from the interval. In the context of sparse data, the number of nodes N is inherently small. For small N, the Runge phenomenon is not a dominant factor, and the stability difference between node sets for in-domain interpolation is less critical. The primary concern becomes the behavior of the extrapolant, which is governed by the growth of $|\omega(x)|$ outside the domain. The choice of equispaced nodes is therefore a pragmatic and theoretically sound decision to control the growth of the extrapolation error.

Our method does not use "custom" weights in an ad-hoc manner; rather, it employs the mathematically correct barycentric weights corresponding to an equispaced grid, as derived from the general formula in Equation 19.

Let the equispaced nodes be $x_j = -1 + (j-1)h$ for $j = 1, \ldots, N$, where the step size is $h = \frac{2}{N-1}$. The term $\omega'(x_j)$ is:

$$\omega'(x_j) = \prod_{k=1,\, k\neq j}^{N} (x_j - x_k) \tag{23}$$

$$= \prod_{k=1,\, k\neq j}^{N} \Big[(-1 + (j-1)h) - (-1 + (k-1)h)\Big] \tag{24}$$

$$= \prod_{k=1,\, k\neq j}^{N} (j-k)h \tag{25}$$

$$= h^{N-1} \prod_{k=1,\, k\neq j}^{N} (j-k) \tag{26}$$

$$= h^{N-1} \cdot (j-1)! \cdot (-1)^{N-j}(N-j)! \tag{27}$$

The barycentric weight ρ_j is the reciprocal:

$$\rho_j = \frac{1}{\omega'(x_j)} = \frac{1}{h^{N-1}(j-1)!(-1)^{N-j}(N-j)!}. \tag{28}$$

We can absorb the constant scaling factor $h^{-(N-1)}$ into the overall normalization of Equation 21, as it cancels out from the numerator and denominator. We can also adjust the alternating sign. A common convention is to define the weights as:

$$\rho_j = (-1)^{j-1}\binom{N-1}{j-1}. \tag{29}$$

This formulation is derived by recognizing that $\frac{(N-1)!}{(j-1)!(N-j)!} = \binom{N-1}{j-1}$ and adjusting the sign and constant factors. The use of these specific, alternating binomial coefficients as weights is therefore not an arbitrary choice but the direct consequence of applying the barycentric principle to an equispaced set of nodes.

In conclusion, the proposed method adopted in Chebyshev-inspired Expolation is a theoretically robust and well-justified technique. Its strength arises from a deliberate set of choices tailored to the problem of extrapolation from sparse data:

1. **Barycentric Formulation:** It leverages the numerical stability and computational efficiency

($O(N)$ per evaluation point) of the barycentric formula (Equation 21).

2. **Equispaced Nodes:** It prioritizes stability in the extrapolation domain ($|x| > 1$) over optimality in the interpolation domain ($|x| \leq 1$). This is justified because the error term $|\omega(x)|$ for equispaced nodes exhibits more controlled growth outside the interval compared to the explosive growth associated with Chebyshev nodes, which is particularly relevant for long-range predictions.
3. **Correct Weights:** It utilizes the mathematically derived barycentric weights for an equispaced grid, $\rho_j \propto (-1)^{j-1}\binom{N-1}{j-1}$, ensuring that the method correctly implements Lagrange polynomial extrapolation in a stable form.

In summary, the method does not contradict established theory but rather applies it judiciously, recognizing that the optimal choice of nodes is context-dependent. For the specified goal of extrapolation from a limited number of points, the combination of equispaced nodes and their corresponding barycentric weights provides a superior and more reliable framework than the traditional Chebyshev-based approach.

B. Pseudocode of ChebBooster

From provided pseudocode in Algorithm 1, we can make it clearer that our ChebBooster algorithm consists of two main stages: an offline precomputation stage and an online application stage, which makes the cache and forcasting process more efficient. The offline precomputation stage is conducted before the inference, where we define a full-computation schedule based on the schedule parameters and precompute the interpolation coefficients for the steps that are not in the full-computation schedule. The online application stage is conducted during the inference, where we initialize an empty history buffer to store the features computed at full-computation steps, and use the pre-computed coefficients to approximate the features at the extrapolation steps.

Algorithm 1 ChebBooster Algorithm

Require: Model M, schedule parameters (s_0, s_1, r), history size n.
1. Offline Precomputation Stage:
Define full-computation schedule $\mathcal{F}$ based on s_0, s_1, r.
For each step $t \notin \mathcal{F}$, pre-compute and store interpolation coefficients $\boldsymbol{w}^t$.
2. Online Application Stage:
Initialize an empty history buffer H (size n).
for each timestep s from 0 to $T-1$ **do**
  if $s \in \mathcal{F}$ **then** ▷ Full computation
    Compute feature f_s using the model M.
    Store the pair (s, f_s) in the history buffer H.
  else ▷ Extrapolation
    Retrieve cached features $\{f_j\}$ from H.
    Retrieve pre-computed coefficients $\boldsymbol{w}^s$.
    Approximate feature $f_s \leftarrow \sum \alpha_j^s \cdot f_j$.
  end if
  Use the feature f_s to proceed with the diffusion step.
end for

C. Additional Introduction to Experiment Settings

Our experiments on DiT-XL/2 and PixArt-Σ are conducted on Nvidia RTX 4090 GPUs, while the experiments on FLUX.1-dev are conducted on an Nvidia A800 GPU. For the experiments of PixArt-Σ, we seperate the whole process into the text embedding process and the sampling process, for the former is conducted redundantly with the same prompts. This step allows us to conduct all of our experiments on only one low-memory GPU. The ImageReward score is evaluated using their official evaluation code, and the evaluation model is ImageReward=1.0, which can be retrieved from the Hugging Face model hub. The ClipScore is evaluated using the code implemented by torchmetrics, and the evaluation model is clip-vit-base-patch32, which can also be retrieved from the Hugging Face model hub. The evaluation code of FID, sFID and Inception Score is from the official implementation of Guided Diffusion. In the C2I task, we set the CFG scale to 1.55, and set the seed to 2025. For

the T2I tasks, we set the seed to 2025 and keep the same settings as the original implementation.

Here we notice there is a mistake in the Reproducibility Checklist, for the Question 4.6, our answer should be "Yes". We apologize for the missing of the answer.

D. Additional Experiments on DiT-XL/2

Method	IS↑	FID↓	sFID↓	FLOPs↓
Original (50 steps)	251.56	2.18	4.29	23.735
Original (25 steps)	244.03	2.75	4.55	11.868
Original (20 steps)	235.35	3.27	4.93	9.494
Original (16 steps)	223.38	4.24	5.69	7.595
Original (12 steps)	195.83	7.07	8.05	5.696
Original (10 steps)	168.99	11.19	11.12	4.747
Cheb ($r = 2, n = 2$)	246.98	2.40	4.91	8.562
Cheb ($r = 3, n = 3$)	246.49	2.25	4.59	8.564
Cheb ($r = 4, n = 4$)	247.27	2.40	4.84	8.566
Cheb ($r = 5, n = 5$)	247.27	2.28	4.64	8.567
Cheb ($r = 2, n = 2$)	244.54	2.62	5.63	6.666
Cheb ($r = 3, n = 3$)	242.58	2.35	4.95	6.668
Cheb ($r = 4, n = 4$)	244.31	2.61	5.44	6.670
Cheb ($r = 5, n = 5$)	244.20	2.38	5.02	6.671
Cheb ($r = 2, n = 2$)	240.99	2.79	5.67	5.717
Cheb ($r = 3, n = 3$)	238.48	2.44	4.96	5.720
Cheb ($r = 4, n = 4$)	241.06	2.75	5.46	5.721
Cheb ($r = 5, n = 5$)	241.26	2.49	4.97	5.722
Cheb ($r = 2, n = 2$)	230.61	3.50	7.68	4.769
Cheb ($r = 3, n = 3$)	228.95	2.87	5.76	4.772
Cheb ($r = 4, n = 4$)	230.38	3.33	6.78	4.773
Cheb ($r = 5, n = 5$)	232.37	2.91	5.85	4.774
Cheb ($r = 2, n = 2$)	223.52	3.98	8.32	4.295
Cheb ($r = 3, n = 3$)	223.52	3.17	5.91	4.297
Cheb ($r = 4, n = 4$)	226.86	3.60	6.94	4.299
Cheb ($r = 5, n = 5$)	228.57	3.14	5.68	4.299

Table 4: **Performance Comparison of ChebBooster on DiT-XL/2.** This table summarizes the performance of ChebBooster across different configurations, comparing Inception Score (IS), FID, sFID, and FLOPs against the original model with varying sampling steps. The results demonstrate that ChebBooster achieves significant speedup while maintaining competitive quality metrics.

We conduct additional experiments on DiT-XL/2 to further validate the effectiveness of ChebBooster. The results are shown in Table 4, and the visualization has been illustrated in Figure 7, Figure 8 and Figure 9. Our evaluation focuses on comparing the proposed ChebBooster method against the Original baseline,

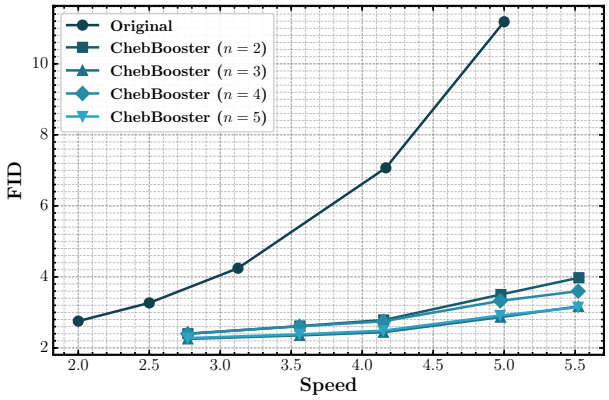

Figure 7: FID evaluation on DiT-XL/2 across varying sampling steps. This figure shows the trade-off between FID scores and inference speed when applying ChebBooster. Notably, ChebBooster achieves substantial acceleration while preserving competitive FID performance, indicating that the generated images remain close to the real data distribution even under aggressive speedup.

performing an ablation study on the hyperparameters of ChebBooster, and offering optimal parameter recommendations based on the empirical results.

Key Observations. Our experiments reveal several critical insights. First, the Original models exhibit a clear and predictable trade-off: reducing inference steps (e.g., from 50 to 10) cuts FLOPs by nearly 80% but causes a catastrophic degradation in generation quality, with the FID score worsening by over 410% (from 2.18 to 11.19). This establishes a critical performance bottleneck for naive acceleration. Second, ChebBooster consistently overcomes this limitation. For instance, `Cheb (r=5, n=5)` achieves a superior FID (2.28 vs. 3.27) and a much higher IS (247.27 vs. 235.35) than `Original (20 steps)` while requiring **10% fewer FLOPs**. This demonstrates that ChebBooster fundamentally improves the performance-efficiency frontier. Finally, the hyperparameters have distinct roles: r primarily governs the foundational acceleration and computational budget, while n acts as a quality-tuning parameter within that budget, consistently improving fidelity for a negligible computational cost.

Parameter Recommendations. Based on this anal-

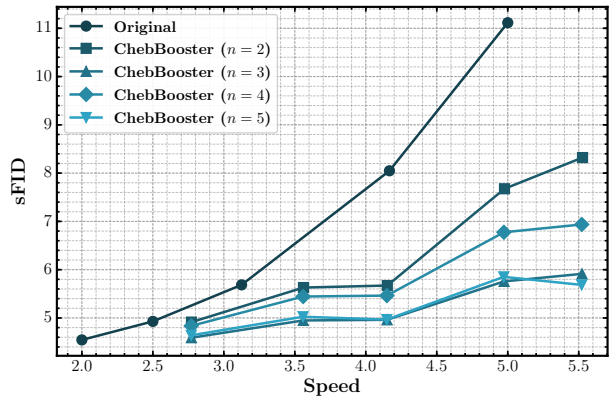

Figure 8: **Structural FID (sFID) evaluation on DiT-XL/2.** The figure highlights how ChebBooster maintains structural consistency in generated images while achieving high speedup ratios. Compared to FID, sFID is more sensitive to localized distortions, and the consistently low sFID values indicate that structural fidelity is preserved even at faster sampling rates.

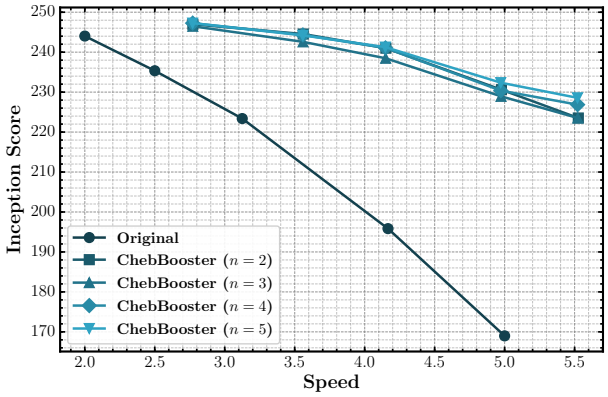

Figure 9: **Inception Score evaluation on DiT-XL/2.** This figure shows how ChebBooster impacts sample diversity and semantic clarity across different acceleration levels. Despite fewer sampling steps, the method sustains high Inception Scores, indicating that the generated samples remain both varied and classifiable.

ysis, we can recommend specific parameter configurations tailored to different use-cases, depending on the desired balance between generation fidelity and computational efficiency.

For Maximizing Generation Quality. If the primary goal is to achieve the best possible output quality while still realizing significant speedup, the optimal choice is **`Cheb (r=3, n=5)`**. It yields the highest Inception Score (247.27) and one of the best FID scores (2.28) among all accelerated models. Its performance is nearly on par with the `Original (50 steps)` baseline (FID of 2.28 vs. 2.18) but reduces the computational cost by over 63%.

For Maximizing Computational Efficiency. If the primary constraint is minimizing FLOPs for deployment on resource-limited hardware, the recommended configuration is **`Cheb (r=7, n=2)`**. It offers the lowest computational footprint (4.295 GFLOPs) of all tested ChebBooster variants. Critically, when compared to the `Original (10 steps)` model which has a similar computational budget, this `ChebBooster` configuration is vastly superior, improving the FID from a poor 11.19 down to a respectable 3.98.

A Balanced Recommendation. For a general-purpose, high-performance setting, a configuration of $r = 4$ with $n \geq 4$ provides an excellent balance. For example, `Cheb (r=4, n=5)` maintains a strong FID of 2.38 and an IS of 244.20, while reducing FLOPs to 6.671—a 72% reduction from the full model.

In summary, ChebBooster is a robust and effective method for accelerating the generative model. By appropriately selecting the acceleration level r and tuning the quality with n, users can achieve performance far superior to naive methods across the entire performance-efficiency spectrum.

E. Extensive Visual Quality Analysis of ChebBooster

To further validate the effectiveness of ChebBooster, we conducted visual comparisons on both class-to-image and text-to-image generation tasks, as shown in Figure 10 and Figure 11. These comparisons include several recent training-free acceleration methods—Δ-DiT, FORA, ToCa, and TaylorSeer—as baselines. Across all evaluated prompts and classes, ChebBooster consistently produces outputs that are visually superior in terms of sharpness, semantic fidelity, and structural coherence.

In the PixArt-Sigma setting (Figure 10), ChebBooster

Figure 10: **Extensive qualitative results on PixArt-Σ.**

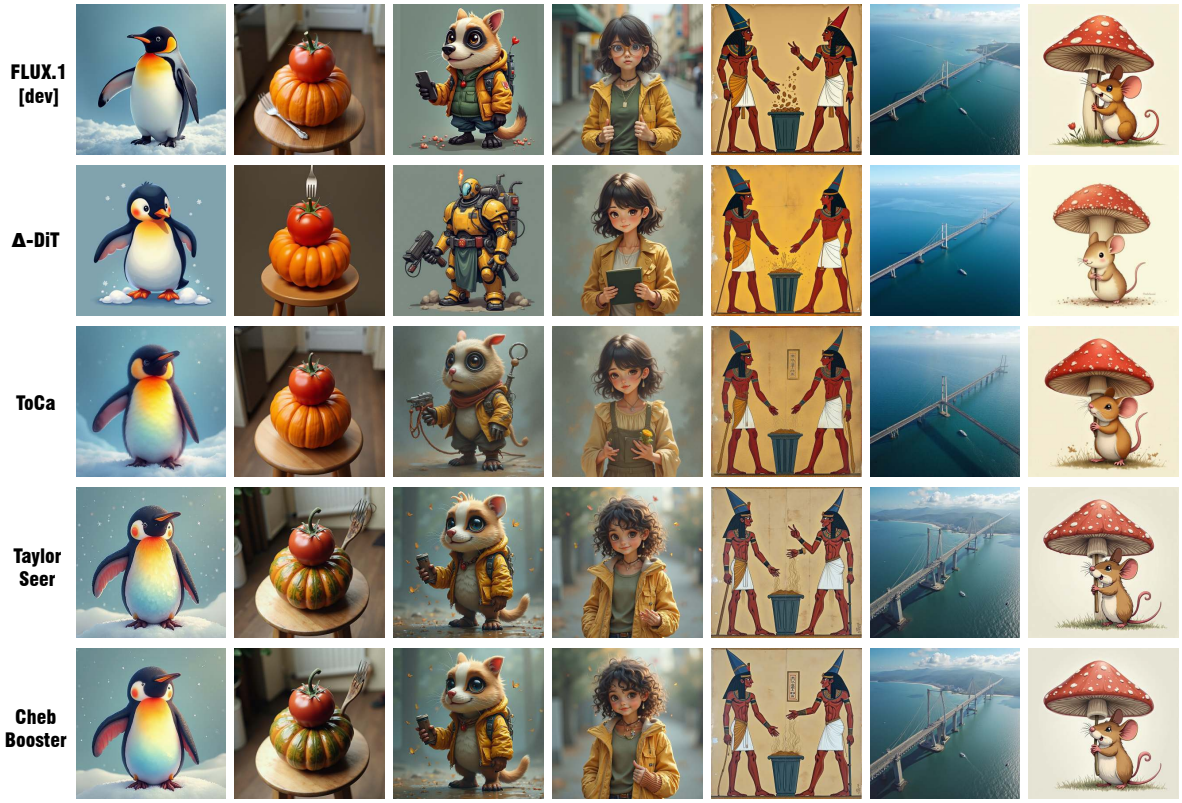

Figure 11: **Extensive qualitative results on FLUX.1-dev.**

avoids the blurriness and semantic degradation exhibited by TaylorSeer and Δ-DiT. While ToCa and FORA generate outputs with acceptable global layout, they often suffer from washed-out textures or oversimplified details. In contrast, ChebBooster maintains fine-grained textures, well-defined contours, and vivid colors, demonstrating its ability to preserve both high-frequency details and global consistency.

Similarly, in the FLUX.1-dev setting (Figure 11), ChebBooster clearly outperforms all other baselines in generating semantically correct and visually appealing images. Notably, ChebBooster retains distinctive visual attributes (e.g., facial features, object poses, and stylistic elements) that are often lost or distorted in other methods. This indicates that the Chebyshev-inspired extrapolation strategy employed by ChebBooster not only improves sampling efficiency, but also enhances the model's ability to maintain feature consistency across denoising steps.

Overall, these qualitative results provide strong evidence that ChebBooster achieves a better trade-off between acceleration and image quality. Its robustness across different models and prompts further suggests that ChebBooster generalizes well to diverse generative scenarios.

F. Prompts for Demonstration

We provide the prompts used in our demonstration of ChebBooster on PixArt-Σ and FLUX.1-dev in Figure 5 and 6. These prompts are designed to showcase the capabilities of ChebBooster in generating high-quality images with various styles and subjects.

1. An Ultraman preparing to take flight. (Figure 5)
2. A full body shot of an elegant, Scottish woman wearing a dress with a sharp focus on her striking eyes in a realistic and beautifully retouched art piece by Artgerm and Jason Chan. (Figure 5)
3. A portrait painting of a red-haired, smiling woman in a green dress against a golden background with intricate patterns. (Figure 5)
4. Yoshitaka Amano's painting of a young lion beastman with a white mane, wearing complex fantasy clothing and huge paws, at a medieval market on a windy day. (Figure 5)
5. A white toilet sitting next to a large window. (mistaken for "A white toilet tin a bathroom sitting next to a sink." in Figure 5)
6. Two people standing in a kitchen near a stove. (Figure 5)
7. A single clock is sitting on a table. (Figure 6)
8. An illustration of a small green elephant standing behind a large red mouse. (Figure 6)
9. Medium shot black and white manga pencil drawing with a highly detailed face of Alita by Yukito Kishiro. (Figure 6)
10. Mr. Bean featured on a WWII propaganda poster holding a gun. (Figure 10)
11. The image portrays Ophelia with a detailed and elegant face, featuring wonderful eyes, wearing an intricate dress, and created with hyperrealistic painting techniques. (Figure 10)
12. A pirate with a beer is illustrated in detailed digital painting. (Figure 10)
13. Rainbow coloured penguin. (Figure 11)
14. A tomato has been put on top of a pumpkin on a kitchen stool. There is a fork sticking into the pumpkin. The scene is viewed from above. (Figure 11)
15. Rbefraigerator. (Figure 11)
16. Matutinal. (Figure 11)
17. An ancient Egyptian painting depicting an argument over whose turn it is to take out the trash. (Figure 11)
18. A bridge connecting Europe and North America on the Atlantic Ocean, bird's eye view. (Figure 11)
19. Illustration of a mouse using a mushroom as an umbrella. (Figure 11)