LESA: Learnable Stage-Aware Predictors for Diffusion Model Acceleration

Peiliang Cai[*] Jiacheng Liu[*] Haowen Xu, Xinyu Wang, Chang Zou, Linfeng Zhang [†]

Shanghai Jiao Tong University

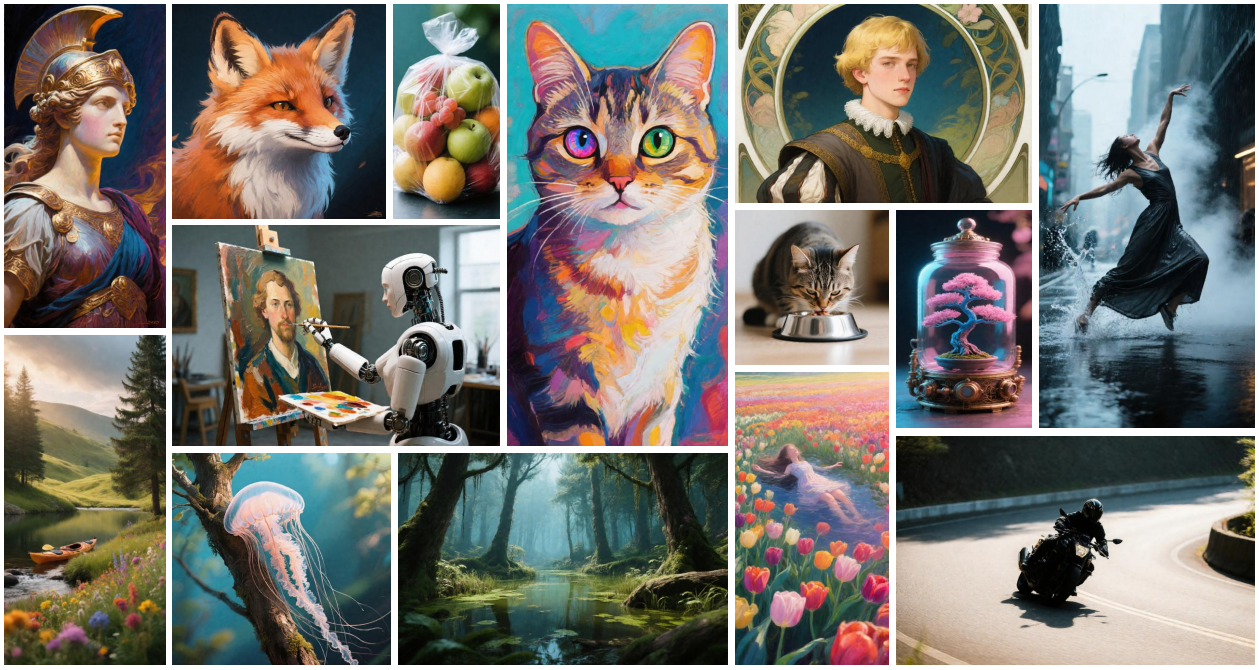


Figure 1. Images generated by Qwen-image using *LESA* with a 6.25× acceleration.

Abstract

*Diffusion models have achieved remarkable success in image and video generation tasks. However, the high computational demands of Diffusion Transformers (DiTs) pose a significant challenge to their practical deployment. While feature caching is a promising acceleration strategy, existing methods based on simple reusing or training-free forecasting struggle to adapt to the complex, stage-dependent dynamics of the diffusion process, often resulting in quality degradation and failing to maintain consistency with the standard denoising process. To address this, we propose a **LE**arnable **S**tage-**A**ware (**LESA**) predictor framework based on two-stage training. Our approach leverages a Kolmogorov–Arnold Network (KAN) to accurately learn temporal feature mappings from data. We further introduce a multi-stage, multi-expert architecture that assigns specialized predictors to different noise-level stages, enabling more precise and robust feature forecasting. Extensive experiments show our method achieves significant acceleration while maintaining high-fidelity generation. Experiments demonstrate 5.00× acceleration on FLUX.1-dev with minimal quality degradation (1.0% drop), 6.25× speedup on Qwen-Image with a 20.2% quality improvement over the previous SOTA (TaylorSeer), and 5.00× acceleration on HunyuanVideo with a 24.7% PSNR improvement over TaylorSeer. State-of-the-art performance on both text-to-image and text-to-video synthesis validates the effectiveness and generalization capability of our training-based framework across different models. Our code is available at* https://github.com/caipeiliang2004/LESA.

1. Introduction

Diffusion Models (DMs)[1, 7, 27] have achieved remarkable success in image and video generation tasks. Recently, Diffusion Transformers (DiTs) [26] have further improved the

[*]Equal contribution.
[†]Corresponding author.

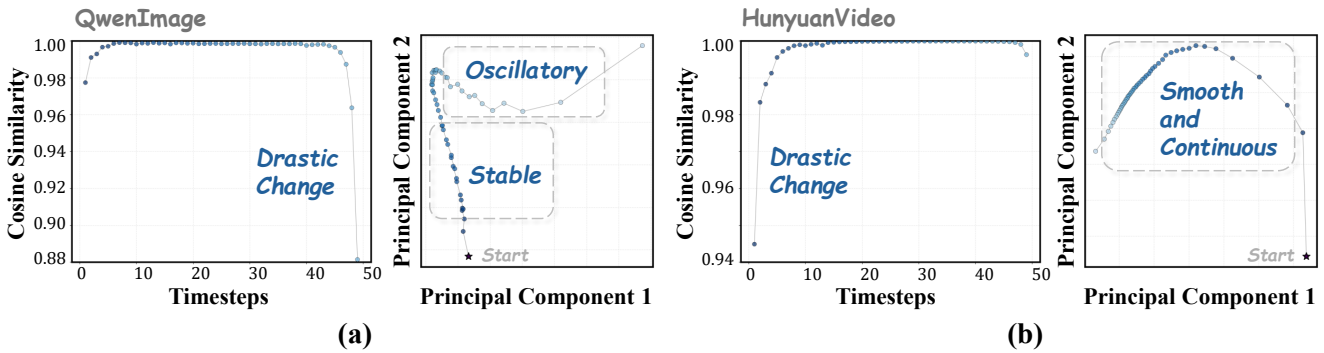


Figure 2. **Cosine Similarity curves and PCA-projected trajectories** show that feature evolution differs markedly across diffusion models, with non-smooth, stage-dependent dynamics that challenge the common assumption of simple or continuous temporal change.

quality and diversity of generated content. However, DiT models face significant computational challenges during inference due to their deep transformer architectures and the need for iterative denoising across multiple timesteps. These limitations restrict their practical deployment in real-world applications. To address this efficiency problem, researchers have proposed a new paradigm known as Feature Caching, which exploits temporal redundancy between consecutive denoising steps to accelerate model inference [2, 12, 16, 25].

Current feature caching methods can be divided into two categories: **"Cache-Then-Reuse"** and **"Cache-Then-Forecast"**. The former relies on the assumption that feature variations between adjacent timesteps in diffusion models are minimal, and directly reuses the features from the previous timestep [31, 48]. In contrast, the latter assumes that *feature evolution of the diffusion process is smooth over time* and uses*training-free* techniques, such as Taylor expansion, to extrapolate features for future timesteps [17]. Although forecasting methods have achieved significant acceleration, their core assumption that features evolve smoothly over time does not always hold in the actual diffusion process.

In our analysis, we observe that the feature dynamics of diffusion models exhibit a ***stage-dependent*** property as shown in Fig. 2. The diffusion process starts with a high-noise stage characterized by drastic and unstable feature changes, and ends with a detail-refinement stage where fine-grained features are progressively reconstructed. In contrast, the middle stage shows more stable and continuous feature evolution. Consequently, conventional caching strategies that adopt ***a single fixed prediction scheme*** across all timesteps are inherently *incapable of adapting to the stage-varying dynamics of the diffusion process*, leading to degraded generation quality and temporal inconsistency.

To address these issues, several time-aware caching strategies have been proposed. Methods like TeaCache [14] and SpeCa [18] adaptively adjust the caching interval to better accommodate the non-uniform dynamics of the diffusion process. Despite these methods enhancing the flexibility of the caching mechanism to some extent, their core still relies on *heuristic rules* and lacks the capacity to learn the underlying temporal dynamics.

To this end, we propose a ***training-based stage-aware learnable predictor framework*** that explicitly models temporal characteristics of the diffusion process for efficient and stable inference acceleration. At its core, we introduce a Kolmogorov–Arnold Network (KAN) [20] as the foundational architecture for the learnable predictor, combining strong learning capacity with mathematical elegance and remarkable parameter efficiency.

Furthermore, we propose ***a multi-stage, multi-expert prediction mechanism***. This framework partitions the diffusion process into distinct stages based on noise levels, deploying a dedicated expert predictor for each. Specifically, a high-noise expert handles the initial stage of drastic feature change; a mid-noise expert manages the stable, continuous denoising in the middle phase; and a low-noise expert is solely responsible for final detail refinement. This architecture explicitly models the stage-dependent nature of the process, ensuring high-quality generation and temporal consistency even at high acceleration ratios.

Correspondingly, the training procedure also follows a staged strategy. It begins with ***ground-truth guided training (GT-Guided Training)*** to learn denoising dynamics, and then progresses to ***closed-Loop autoregressive training (CL-AR Training)*** to build robustness against the inevitable prediction drift encountered during fast, multi-step inference.

Our main contributions are as follows:

- **Learnable Temporal Prediction Framework:** We introduce a parameter-efficient predictor based on a Kolmogorov-Arnold Network that directly learns temporal dynamics from diffusion model activations, overcoming the limitations of training-free forecasting methods.
- **Stage-Aware Multi-Expert Architecture:** Our framework partitions the denoising process into distinct stages, each handled by a dedicated expert predictor. This specialized design ensures accurate modeling of the unique feature dynamics at different noise levels, enhancing both prediction fidelity and generalization.
- **Two-Stage Training Strategy:** We propose a two-stage training methodology that begins with ground-truth guided

training to learn denoising dynamics and progresses to closed-loop autoregressive training to build robustness against prediction drift, ensuring stable performance during accelerated inference.

2. Related Works

Diffusion models have shown remarkable generative performance across various tasks, like image and video synthesis. As research progresses, their architectures have evolved from convolutional U-Nets [28] to diffusion transformers [26], shifting from local feature modeling to global context modeling and achieving stronger representation and scalability. However, the improved generation quality of large-scale diffusion models [10, 13, 35, 40] comes at the cost of heavy computation: each denoising process requires dozens of complex network evaluations, leading to high latency and demanding hardware resources. To address this bottleneck, researchers have focused on two main directions: *reducing temporal complexity* and *improving per-step efficiency*.

2.1. Reducing Temporal Iteration

Temporal reduction can be approached from two perspectives: *sampling schedulers* and *trajectory compression*.

Sampling-based Schedulers. Sampling-based acceleration reduces inference steps while preserving image quality. DDIM [32] introduced deterministic sampling with fewer iterations, and the DPM-Solver [22, 23] series further improved efficiency using high-order ODE solvers. Flow Matching [19] later generalized diffusion into a deterministic transformation between noise and data, enabling faster and more stable sampling. However, their performance is highly dependent on model architecture and noise schedule, making generalization across different frameworks difficult.

Trajectory-based Compression. Instead of repeatedly simulating the entire diffusion trajectory, this line of work trains models to approximate the multi-step denoising behavior in a limited number of steps. Progressive compression [39] techniques enable few-step inference, while Consistency Models [33] achieve nearly one-step generation by enforcing trajectory consistency across timesteps. However, these methods require additional training, and their acceleration gains are relatively limited compared with other approaches.

2.2. Enhancing Efficiency in Denosing-Step

Beyond temporal reduction, another direction aims to enhance per-step efficiency by minimizing redundant computation and memory usage while preserving generation quality.

Model Simplification and Token Reduction. Considering the continuous scaling up of model size, some works [3, 8, 46] aim to simplify the model's architecture to reduce computational cost in each inference. Another line of research focuses on reducing the number of tokens fed into the diffusion models, thereby accelerating attention computation. Token pruning [42] and adaptive token selection [29] methods dynamically discard visually or semantically redundant tokens while retaining essential contextual information for generation. However, aggressive compression easily breaks feature consistency and harms visual fidelity, making it difficult to accelerate without degrading output quality.

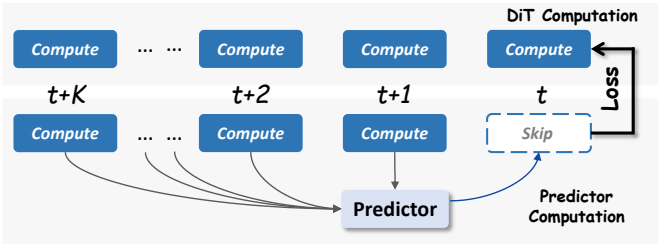
Feature Caching and Reuse Mechanisms. Feature caching, as a relatively popular research direction in recent years, has attracted attention due to its characteristics of training-free and effectiveness. Feature caching [25] was initially introduced in the diffusion model based on U-Net, and then applied to solve the timestep redundancy of DiTs [2, 31]. Early static cache methods focused on the implementation based on scheduling [24] for feature caching with timestep and model layer customization, exploring redundancy calculations in spatial [18, 45, 48], temporal [47], and conditional dimensions [41]. With the introduction of dynamic feature [9, 14, 18] update methods, this sample-adaptive feature caching further expanded its effectiveness. Furthermore, with the introduction of predictive caching by TaylorSeer [17], recent work [4, 15, 44] has explored caching predictions based on different polynomial methods to achieve approximate estimation of features, thereby realizing higher acceleration in diffusion inference.

While predictive methods achieve *impressive acceleration*, they often suffer from *poor consistency* with the original image or video. Large caching intervals without correction easily distort spatial structures, as *simple polynomial methods* fail to capture the *complex, coupled dynamics* of diffusion. These limitations motivate a *learnable* predictor that models temporal evolution more accurately through a *stage-wise* design tailored to different diffusion phases.

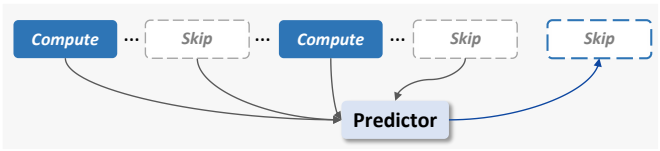
2.3. Kolmogorov-Arnold Network

Kolmogorov–Arnold Network (KAN)[20] is a neural architecture grounded in the Kolmogorov–Arnold representation theorem, which states that *any multivariate continuous function can be expressed through compositions and summations of univariate functions*. Unlike MLP with fixed activations, KAN uses learnable spline-based univariate functions at each neuron, providing flexible nonlinear modeling. Subsequent studies [5, 21, 34] have demonstrated their effectiveness in diverse time-series prediction tasks, confirming their ability to capture complex temporal relationships. With strong functional expressiveness and adaptive approximation capability, KAN serves as an ideal learnable predictor for modeling the nuanced dynamics of the diffusion process.

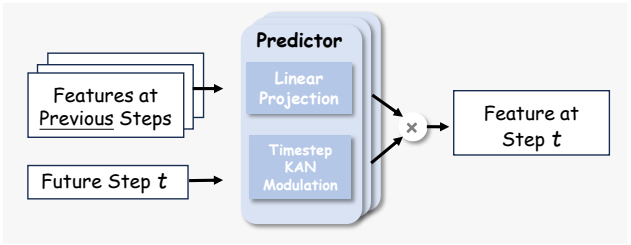
These insights motivate us to adopt KAN as a learnable predictor to correct the drift present in training-free caching. When integrated with our stage-aware modeling of diffusion dynamics, this yields a learnable-cache paradigm for accelerating diffusion inference.



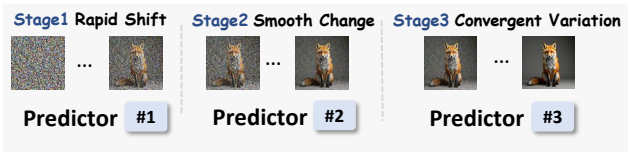
(a) Training Progress of the Predictor



(b) Inference Process with the Predictor



(c) Architecture of the Predictor



(d) Different Phrases Use Different Predictors

Figure 3. **Overview of the training-based stage-aware learnable predictor framework. (a) Training progress** of the *LESA* uses outputs from the previous K steps as input and updates the predictor with the DiT activation output. **(b) Inference process** with the learned predictor skips part of the DiT computations by predicting features. **(c) Predictor architecture** processes features from previous steps with a linear projection and processes the timestep with a KAN module, whose outputs are multiplied to generate the feature at step t. **(d) Stage-aware denoising** divides the trajectory into three stages and assigns a dedicated predictor to each stage.

3. Method

3.1. Preliminary

Feature Caching in Diffusion Models Feature caching aims to accelerate diffusion inference by avoiding redundant computation across adjacent timesteps. In the conventional *"cache-then-reuse"* paradigm, features computed at timestep t are directly stored and reused for subsequent steps, i.e.,

$$\mathcal{F}(x^l_{t-k}) \approx \mathcal{F}(x^l_t), \quad k \in 1, \ldots, N-1 \tag{1}$$

This naive reuse offers an $(\mathcal{N}-1)$-fold theoretical speedup but neglects temporal feature evolution, leading to rapid error accumulation as $\mathcal{N}$ grows. To mitigate error accumulation, the recent *"cache-then-forecast"* paradigm reformulates feature caching as a forecasting problem, exploiting the smooth and continuous nature of feature trajectories in diffusion models. Instead of copying features, it models temporal feature evolution using finite differences or polynomial extrapolation. The predicted feature at timestep $t-k$ can be expressed as a truncated Taylor expansion:

$$\mathcal{F}_{\text{pred},m}(x^l_{t-k}) = \mathcal{F}(x^l_t) + \sum_{i=1}^{m} \frac{\Delta^i \mathcal{F}(x^l_t)}{i! \cdot N^i}(-k)^i, \tag{2}$$

where $\Delta^i \mathcal{F}(x^l_t)$ denotes the i-th finite difference. This formulation transforms caching from simple reuse to temporal prediction, enabling accurate feature estimation.

Kolmogorov-Arnold Network KAN is built upon the Kolmogorov–Arnold representation theorem, which states that any multivariate continuous function can be expressed using only univariate functions composed with linear projections. Following this principle, a generic KAN layer realizes the mapping $\mathbf{x} \mapsto \mathbf{y}$ in the form

$$\mathbf{y} = \sum_{m=1}^{M} \mathbf{w}_m \, \phi_m\big(\mathbf{a}_m^\top \mathbf{x}\big), \tag{3}$$

where $\mathbf{a}_m$ and $\mathbf{w}_m$ parameterize the inner and outer linear mappings, and each $\phi_m(\cdot)$ is a learnable spline-based univariate function. Compared with networks using fixed nonlinearities, KAN explicitly represents functions as combinations of such univariate components, which improves its ability to approximate complex smooth dynamics.

3.2. LESA Framework

Stage-Aware Temporal Segmentation The dynamics of the diffusion denoising trajectory are inherently non-uniform: As shown in Fig. 3 (d), the high-noise regime is dominated by coarse feature formation, whereas the low-noise regime emphasizes fine-grained detail refinement. To accurately capture these stage-dependent dynamics, we divide the denoising process into discrete stages, each handled by a specialized sub-model tailored to its distinctive behavior. Furthermore, the temporal window K is likewise adapted to each stage. A shorter window (K=4) is employed in the high-noise phase to accommodate its rapid and weakly correlated transitions, whereas a longer window (K=8) is utilized in the subsequent, more stable phases to better capture the intricate, history-dependent evolution of features.

This stage-aware design enables the *LESA* to leverage specialized temporal modeling strategies that align with the statistical and structural properties of each generation stage.

KAN-based Temporal Modeling To model the nonlinear temporal dependency from cached features $\{\mathbf{h}_{t+K-1}, \mathbf{h}_{t+K-2}, \ldots, \mathbf{h}_t\}$ to the target feature $\mathbf{h}_{t-1}$, we

design a *compact residual LESA predictor* that combines a linear feature projection with a Kolmogorov–Arnold Network (KAN) for timestep modulation as illustrated in Fig. 3 (c). Specifically, the cached feature sequence is linearly projected to obtain a feature representation:

$$\mathbf{z} = \mathbf{W}[\mathbf{h}_{t+K-1}, \mathbf{h}_{t+K-2}, \ldots, \mathbf{h}_t] + b, \tag{4}$$

where $\mathbf{W}$ and b are learnable parameters, and $[\cdot]$ denotes concatenation. Meanwhile, we compute the relative temporal offsets between the prediction step and all timesteps within the full sequence of L timesteps as $\Delta t_i = t_{\text{pred}} - t_i$, for $i = L-1, \ldots, 0$, and feed them into the KAN to produce a scalar temporal modulation factor:

$$\alpha = f_{\text{KAN}}(\Delta t_{L-1}, \ldots, \Delta t_0). \tag{5}$$

Let $\Delta \mathbf{t} = [\Delta t_{L-1}, \ldots, \Delta t_0]^\top$ be the relative timestep vector. Following the general KAN form in Eq. 3, we write

$$\alpha = f_{\text{KAN}}(\Delta \mathbf{t}) = \sum_{m=1}^{M} w_m \, \phi_m\big(a_m^\top \Delta \mathbf{t}\big), \tag{6}$$

where a_m and w_m are learnable coefficients, and each $\phi_m(\cdot)$ is a smooth spline-based function. This yields a flexible scalar function $\alpha = g(t_{\text{pred}})$ that modulates the residual update strength: the predictor learns larger α in early high-noise steps and smaller values during late refinement.

The scalar factor is then broadcast-multiplied with the feature to form the residual update:

$$\hat{\mathbf{h}}_{t-1} \;=\; \mathbf{h}_t \;+\; \alpha\, \mathbf{z}, \tag{7}$$

where α modulates all feature dimensions uniformly across channels. This formulation follows a separation-of-variables principle: The linear projection transforms the feature sequence in space, while the KAN models a continuous scalar temporal modulation over the entire L-step sequence.

Training Procedure. Our training procedure consists of three stages: *Data Preparation*, *Ground-Truth Guided Training*, *Closed-Loop Autoregressive Training*.

We first run the base diffusion model without acceleration and cache its trajectories. For each prompt, we save the initial noise and the feature at all denoising timesteps.

In the first stage, *LESA* is trained under a ***ground-truth guided regime***. At each timestep t, as illustrated in Fig. 3(a), it receives the past K GT features along with the timestep embedding, and is optimized by minimizing the L_1 loss between its prediction and the GT feature at timestep t.

In the second stage, we adopt a ***closed-loop autoregressive training scheme*** that mirrors real inference. The predictor now conditions on the past K features composed of both its own historical predictions and the model's actual outputs. This exposure to accumulated drift enables the predictor to learn how features evolve under imperfect inputs, resulting in more robust behavior during accelerated inference.

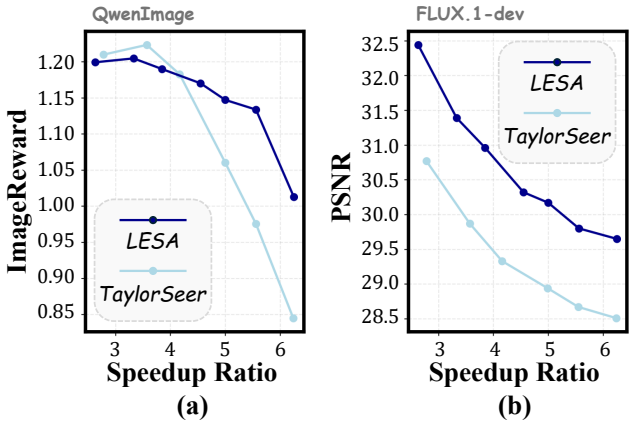


Figure 4. **Comparison of *LESA* and TaylorSeer under different speedup ratios.** *LESA* preserves better quality metrics and perceptual metrics On QwenImage and FLUX.

For a comprehensive description of the training details, please refer to the supplementary material.

4. Experiments

Evaluation and Metrics For the text-to-image task, we evaluate on DrawBench[30] using ImageReward[38] and CLIP Score[6] as quality metrics, and PSNR, SSIM[36], and LPIPS[43] as perceptual metrics. For the text-to-video task, we use the same perceptual metrics to assess frame-wise fidelity. We highlight the **best results in bold** and underline our method when it ranks second. *See the supplementary material for configuration and evaluation details, along with additional experiments on distilled models.*

4.1. Text-to-Image Generation

4.1.1. FLUX.1-dev and FLUX.1-schnell

Quantitative Study We evaluate *LESA* against existing acceleration methods under comparable settings (shown in Tab. 1). In the moderate-acceleration regime, *LESA* ($\mathcal{N}$=5) achieves a 3.85× speedup with a PSNR of 30.96, outperforming TaylorSeer ($\mathcal{N}$=4, O=2) and TeaCache (l=0.6) in both efficiency and perceptual quality. As acceleration increases, *LESA* maintains a 5.00× speedup with a CLIP Score of 32.88 and PSNR of 30.17. Even at the highest acceleration, *LESA* sustains a 6.25× speedup with an ImageReward of 0.91, PSNR of 29.65, and LPIPS of 0.40, demonstrating strong robustness against quality collapse.

Furthermore, on FLUX.1-schnell, *LESA* maintains strong performance even under extreme step reduction like $\mathcal{N}$=4, achieving substantial acceleration with only a modest quality drop. Notably, when distilled models behave suboptimally, the cache-based compression of intermediate representations can even improve stability and visual fidelity, further highlighting the practicality of our approach.

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

Method	Acceleration				Quality Metrics		Perceptual Metrics		
FLUX	**Latency(s)** ↓	**Speed** ↑	**FLOPs(T)** ↓	**Speed** ↑	**Image Reward**↑	**CLIP Score**↑	**PSNR**↑	**SSIM**↑	**LPIPS**↓
[dev]: 50 steps	23.24	1.00×	3726.87	1.00×	0.99	32.64	∞	1.00	0.00
50% **steps**	11.82	1.97×	1863.44	2.00×	0.97	32.57	29.56	0.73	0.31
PAB	17.84	1.30×	3013.13	1.24×	0.95	32.55	28.84	0.67	0.40
DBCache	16.88	1.38×	2384.29	1.56×	1.01	32.53	33.86	0.87	0.12
FORA ($\mathcal{N}$=3)	9.06	2.57×	1267.89	2.94×	0.99	32.27	30.65	0.77	0.25
TeaCache (l=0.6)	9.13	2.55×	1342.20	2.78×	0.91	32.11	29.03	0.68	0.40
TaylorSeer ($\mathcal{N}$=4, O=2)	8.78	2.64×	1118.86	3.57×	**1.02**	32.41	29.87	0.73	0.30
LESA ($\mathcal{N}$=5)	**6.57**	**3.54**×	**969.11**	**3.85**×	1.00	**32.70**	**30.96**	**0.77**	**0.24**
FORA($\mathcal{N}$=5) †	5.97	3.89×	820.80	4.54×	0.82	32.48	28.44	0.60	0.50
ToCa($\mathcal{N}$=8, $\mathcal{R}$=75%)	12.39	1.88×	829.86	4.49×	0.95	32.60	29.07	0.64	0.43
DuCa($\mathcal{N}$=8, $\mathcal{R}$=70%)	9.40	2.47×	858.27	4.34×	0.94	32.58	29.06	0.64	0.43
TeaCache(l=1.0) †	7.07	3.29×	820.55	4.54×	0.84	31.88	28.61	0.64	0.48
TaylorSeer($\mathcal{N}$=6, O=2)	6.73	3.45×	746.28	4.99×	**1.02**	32.53	28.94	0.66	0.40
LESA ($\mathcal{N}$=7)	**5.19**	**4.48**×	**745.51**	**5.00**×	0.98	**32.88**	**30.17**	**0.72**	**0.32**
FORA($\mathcal{N}$=7) †	5.09	4.57×	597.25	6.24×	0.68	31.90	28.32	0.59	0.54
ToCa($\mathcal{N}$=12, $\mathcal{R}$=85%) †	9.82	2.37×	618.57	6.02×	0.80	32.32	28.70	0.60	0.52
DuCa($\mathcal{N}$=12, $\mathcal{R}$=80%) †	7.74	3.00×	646.97	5.76×	0.77	32.20	28.71	0.60	0.53
TeaCache(l=1.4) †	6.14	3.79×	671.51	5.55×	0.74	31.78	28.12	0.48	0.68
TaylorSeer($\mathcal{N}$=9, O=2) †	5.85	3.97×	597.25	6.24×	0.86	32.04	28.38	0.59	0.51
LESA ($\mathcal{N}$=10)	**4.28**	**5.43**×	**596.44**	**6.25**×	**0.91**	**32.65**	**29.65**	**0.67**	**0.40**
[schnell]: 4 steps	2.13	1.00×	278.41	1.00×	0.93	34.09	∞	1.00	0.00
LESA ($\mathcal{N}$=3)	1.31	1.63×	139.21	2.00×	0.95	34.48	30.31	0.80	0.16
LESA ($\mathcal{N}$=4)	1.15	1.85×	69.61	4.00×	0.91	34.67	29.21	0.69	0.30

- † Methods exhibit significant degradation in image quality.

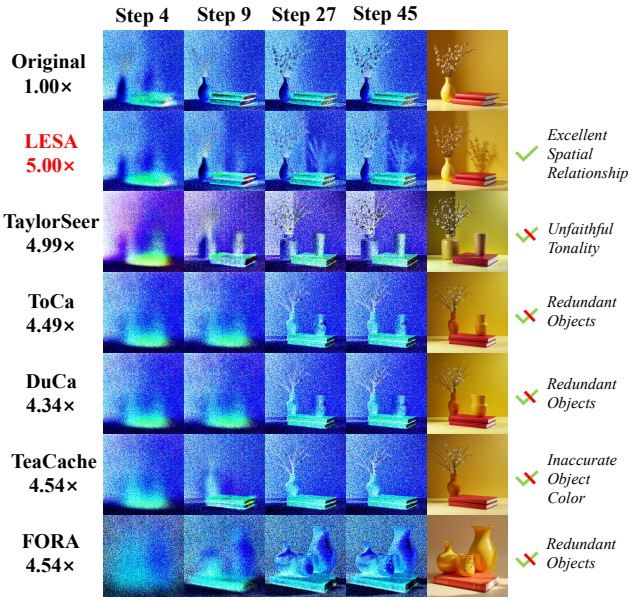


Figure 5. **Comparison of Sampling Process Image and Final Image.** On FLUX, *LESA* demonstrates significant speedup ratio with excellent spatial organization and accurate color representation.

Qualitative Study As shown in Fig. 5, *LESA* achieves a 5.00× speedup while preserving spatial structure, color consistency, and alignment with the target image throughout sampling. In contrast, TaylorSeer shows drift and structural distortion despite similar acceleration, and other methods generate inconsistent objects that break semantic coherence.

4.1.2. Qwen-Image and Qwen-Image-Lightning

Quantitative Study We evaluate our approach on Qwen-Image and its distilled variant, Qwen-Image-Lightning. As shown in Tab. 2, *LESA* ($\mathcal{N}$=7) achieves a 5.00× speedup with **high image consistency**, clearly outperforming other methods that typically fall below 30 PSNR and above 0.40 LPIPS. With a stronger acceleration setting, *LESA* ($\mathcal{N}$=10) reaches 6.25× speedup while still maintaining competitive fidelity, whereas baseline methods experience significant quality degradation. On Qwen-Image-Lightning, *LESA* achieves up to 4× speedup while consistently preserving high image quality. Even at extreme step setting, it remains close to the original baseline in perceptual fidelity.

Qualitative Study Across the visual comparison in Fig. 6, *LESA* at 6.25× delivers results closest to the original image. It preserves whale anatomy and lighting, and clearly retains the latte foam lettering, while other methods exhibit global appearance shifts and noticeable loss of details.

4.2. Text-to-Video Generation

Quantitative Study As shown in Tab. 3, directly reducing the number of steps brings notable speedup but results in a

Table 2. **Quantitative comparison in text-to-image generation** for Qwen-Image and Qwen-Image-Lightning.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
Qwen	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
50 **steps**	127.40	1.00×	12917.56	1.00×	1.25	35.59	∞	1.00	0.00
20% **steps** †	25.92	4.89×	2583.51	5.00×	0.94	34.95	28.59	0.61	0.52
FORA($\mathcal{N}$=4)†	38.43	3.32×	3359.99	3.84×	0.93	34.40	28.66	0.59	0.51
ToCa($\mathcal{N}$=8, $\mathcal{R}$=75%)	61.37	2.08×	2991.34	4.32×	1.02	34.96	28.93	0.63	0.44
DuCa($\mathcal{N}$=9, $\mathcal{R}$=80%)†	34.73	3.67×	2958.13	4.37×	0.77	34.62	28.45	0.58	0.55
TaylorSeer($\mathcal{N}$=6, O=2)	30.29	4.21×	2585.46	5.00×	1.01	34.71	28.58	0.62	0.46
LESA ($\mathcal{N}$=7)	**27.34**	**4.66×**	**2584.90**	**5.00×**	**1.15**	**35.36**	**30.18**	**0.78**	**0.25**
FORA($\mathcal{N}$=6)†	28.69	4.44×	2326.74	5.55×	0.48	33.34	28.48	0.55	0.59
ToCa($\mathcal{N}$=12, $\mathcal{R}$=85%)†	50.95	2.50×	2406.20	5.37×	0.55	34.08	28.69	0.57	0.53
DuCa($\mathcal{N}$=12, $\mathcal{R}$=90%)†	28.57	4.46×	2171.56	5.95×	0.41	33.38	28.38	0.57	0.60
TaylorSeer($\mathcal{N}$=8, O=2)†	25.24	5.05×	2068.86	6.24×	0.84	33.77	28.14	0.47	0.68
LESA ($\mathcal{N}$=10)	**22.40**	**5.69×**	**2068.03**	**6.25×**	**1.01**	**34.93**	**29.23**	**0.73**	**0.34**
[Lightning]: 8 steps	4.37	1.00×	560.96	1.00×	1.29	35.32	∞	1.00	0.00
LESA ($\mathcal{N}$=2)	2.95	1.48×	350.61	1.60×	1.28	35.20	33.42	0.85	0.10
LESA ($\mathcal{N}$=3)	2.48	1.76×	280.49	2.00×	1.27	35.24	31.38	0.80	0.16
LESA ($\mathcal{N}$=4)	1.52	2.88×	140.25	4.00×	1.23	35.66	28.85	0.63	0.37

- † Methods exhibit significant degradation in image quality.

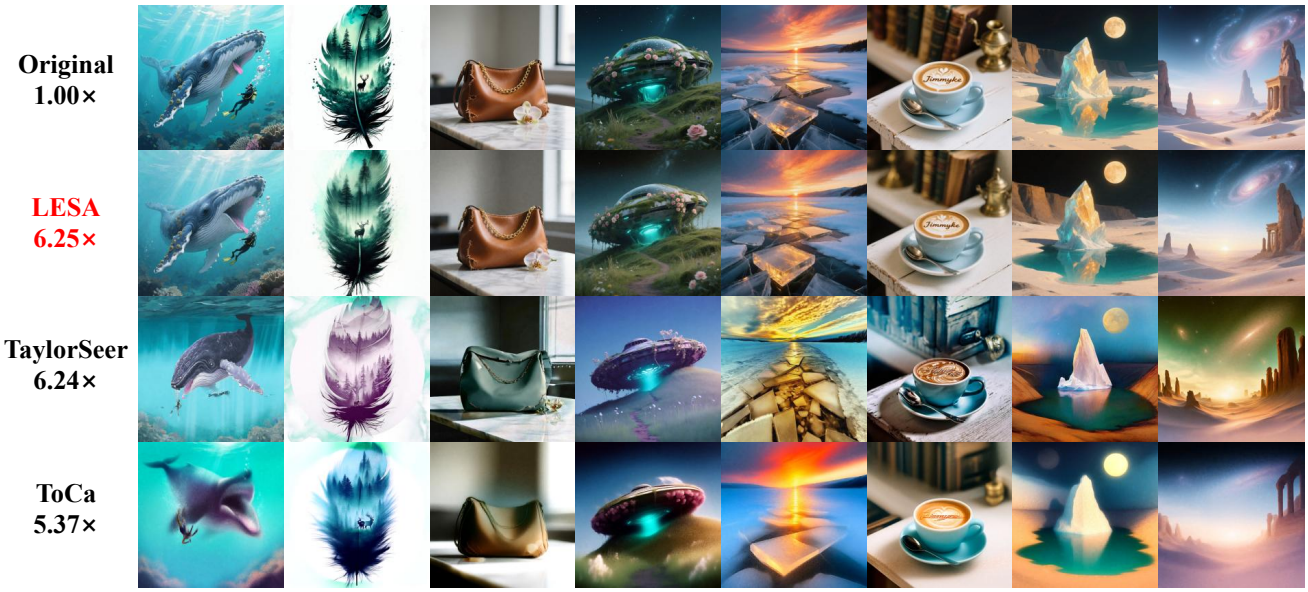


Figure 6. **Qualitative Comparison of Generated Images.** On Qwen-Image, *LESA* delivers higher speedup with high image quality.

substantial degradation of perceptual quality, and existing baselines exhibit a similar trade-off. In contrast, *LESA* simultaneously achieves faster inference and markedly better visual fidelity. With $\mathcal{N}$=7, it attains the best overall performance, improving PSNR by **23.9%** over TaylorSeer; even under the more aggressive $\mathcal{N}$=8 setting, it still provides the highest acceleration while retaining strong perceptual quality, with a **21.7%** PSNR gain.

Qualitative Study As shown in Fig. 7, *LESA* at 5.56× preserves scene semantics and temporal consistency more reliably than all baselines. For *banana on the bottom of an apple*," it keeps the object placement stable, while others distort shapes or shift positions. In "*ink swirling in water*," it maintains multicolor flow and smooth dynamics, whereas competing methods lose color fidelity or introduce artifacts.

4.3. Ablation Study

As detailed in Tab. 4, our ablation study investigates the impact of the inference interval $\mathcal{N}$, stage segmentation, and the timestep modulation module. The results consistently demonstrate that the combination of stage segmentation with a KAN-based module yields superior performance. This

Table 3. **Quantitative comparison of text-to-video generation** on HunyuanVideo.

Method HunyuanVideo	Acceleration				Perceptual Metrics		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	PSNR↑	SSIM↑	LPIPS↓
Original: 50 steps	98.91	1.00×	29773.0	1.00×	∞	1.00	0.00
22% **steps**	22.98	4.30×	6550.1	4.55×	17.65	0.59	0.42
ToCa($\mathcal{N}=5, \mathcal{R}=90\%$)	26.12	3.79×	7006.2	4.25×	17.04	0.54	0.44
DuCa($\mathcal{N}=5, \mathcal{R}=90\%$)	23.08	4.29×	6483.2	4.48×	17.08	0.54	0.43
TeaCache($l=0.4$)	21.83	4.53×	6550.1	4.55×	18.25	0.61	0.38
FORA($N=5$)	22.61	4.37×	5960.4	5.00×	17.00	0.53	0.44
TaylorSeer($\mathcal{N}=5, O=1$)	23.66	4.18×	5960.4	5.00×	17.29	0.55	0.42
SpeCa($\mathcal{N}_{\text{max}}=8, \mathcal{N}_{\text{min}}=2$)	23.22	4.26×	5692.7	5.23×	17.73	0.59	0.39
ClusCa($\mathcal{N}=5, O=1, K=16$)	24.35	4.06×	5373.0	5.54×	17.28	0.55	0.42
LESA($\mathcal{N}=7$)	19.37	5.11×	5960.4	5.00×	**21.43**	**0.72**	**0.29**
LESA($\mathcal{N}=8$)	**17.47**	**5.66**×	**5354.9**	**5.56**×	21.05	0.70	0.32

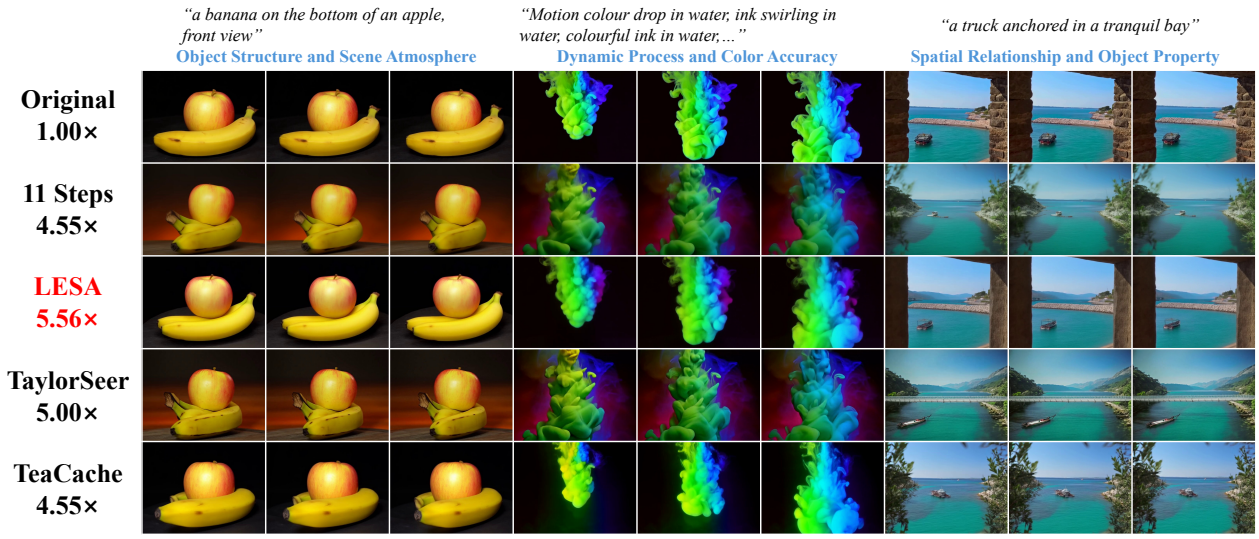


Figure 7. **Qualitative Comparison of Early, Middle, and Late Frames From Generated Videos.** On HunyuanVideo, *LESA* constructs continuous dynamic process with excellent subject accuracy under high speedup ratio.

Table 4. **Ablation study for Interval, Stage Segmentation and Timestep Modulation Model** for *LESA* on FLUX

Experiment Settings			Quality Metrics		Perceptual Metrics		
Interval	Segmented	Module	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
$\mathcal{N}=5$	✔	KAN	1.00	32.70	30.96	0.77	0.24
$\mathcal{N}=5$	✘	KAN	1.00	32.67	30.77	0.77	0.25
$\mathcal{N}=5$	✔	MLP	0.99	32.69	30.76	0.77	0.25
$\mathcal{N}=5$	✘	MLP	0.95	32.50	30.29	0.73	0.31
$\mathcal{N}=7$	✔	KAN	0.98	32.88	30.17	0.72	0.32
$\mathcal{N}=7$	✘	KAN	0.95	32.75	30.10	0.71	0.33
$\mathcal{N}=7$	✔	MLP	0.96	32.72	30.11	0.71	0.33
$\mathcal{N}=7$	✘	MLP	0.95	32.65	30.08	0.71	0.34
$\mathcal{N}=10$	✔	KAN	0.91	32.65	29.65	0.67	0.40
$\mathcal{N}=10$	✘	KAN	0.88	32.58	29.60	0.66	0.41
$\mathcal{N}=10$	✔	MLP	0.89	32.57	29.59	0.66	0.42
$\mathcal{N}=10$	✘	MLP	0.88	32.51	29.55	0.66	0.43

advantage is particularly pronounced at larger intervals, such as $\mathcal{N}$=10, which involve more significant temporal dynamics. In contrast, at smaller intervals ($\mathcal{N}$=5), the high feature similarity between steps leads to only marginal performance differences among the configurations. Crucially, at $\mathcal{N}$=10, our full model not only outperforms the baseline by a significant 3.4% in ImageReward but also achieves concurrent improvements across all perceptual metrics.

5. Conclusion

To address a key limitation in predictable feature caching,namely that *fixed prediction schemes cannot adapt to the stage-dependent dynamics of diffusion models*, we introduce *LESA*, a **stage-aware learnable predictor** built on a two-stage training strategy. By partitioning the denoising trajectory into distinct stages and equipping each with a dedicated KAN-based temporal predictor, *LESA* learns stage-specific temporal feature evolution instead of relying on fixed heuristics. Experiments on image and video generation demonstrate that *LESA* maintains higher perceptual quality and competitive performance even at 6.25× acceleration, validating that modeling stage-specific dynamics is an effective approach for efficient diffusion acceleration.

Acknowledgement

This work was sponsored by WUYING - Alibaba Cloud.

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 1

[2] 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. 2, 3

[3] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *arXiv preprint arXiv:2305.10924*, 2023. 3

[4] Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, and Linfeng Zhang. Hicache: Training-free acceleration of diffusion models via hermite polynomial-based feature caching, 2025. 3

[5] Remi Genet and Hugo Inzirillo. A temporal kolmogorov-arnold transformer for time series forecasting. *arXiv preprint arXiv:2406.02486*, 2024. 3

[6] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 5

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. arXiv:2006.11239 [cs]. 1

[8] Rui Huang, Shitong Shao, Zikai Zhou, Pukun Zhao, Hangyu Guo, Tian Ye, Lichen Bai, Shuo Yang, and Zeke Xie. Diffusion dataset condensation: Training your diffusion model faster with less data. *arXiv preprint arXiv:2507.05914*, 2025. 3

[9] 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, 2024. 3

[10] 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. 3, 2

[11] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 2

[12] Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of unet encoder in diffusion models. *arXiv preprint arXiv:2312.09608*, 2023. 2

[13] Zhimin Li, Jianwei Zhang, and and others Lin. Hunyuan-DiT: A powerful multi-resolution diffusion transformer with fine-grained chinese understanding. 3

[14] 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, 2024. 2, 3

[15] Jiacheng Liu, Peiliang Cai, Qinming Zhou, Yuqi Lin, Deyang Kong, Benhao Huang, Yupei Pan, Haowen Xu, Chang Zou, Junshu Tang, Shikang Zheng, and Linfeng Zhang. Freqca: Accelerating diffusion models via frequency-aware caching, 2025. 3

[16] Jiacheng Liu, Xinyu Wang, Yuqi Lin, Zhikai Wang, Peiru Wang, Peiliang Cai, Qinming Zhou, Zhengan Yan, Zexuan Yan, Zhengyi Shi, et al. A survey on cache methods in diffusion models: Toward efficient multi-modal generation. *arXiv preprint arXiv:2510.19755*, 2025. 2

[17] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers, 2025. 2, 3

[18] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Kaixin Li, Shaobo Wang, and Linfeng Zhang. Speca: Accelerating diffusion transformers with speculative feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25)*, page to appear, Dublin, Ireland, 2025. ACM. 2, 3

[19] Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[20] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *ArXiv*, abs/2404.19756, 2024. 2, 3

[21] Ioannis E Livieris. C-kan: A new approach for integrating convolutional layers with kolmogorov–arnold networks for time-series forecasting. *Mathematics*, 12(19):3022, 2024. 3

[22] 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. 3

[23] 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. 3

[24] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *arXiv preprint arXiv:2406.01733*, 2024. 3

[25] 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. 2, 3

[26] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2023. arXiv:2212.09748 [cs]. 1, 3

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2022. arXiv:2112.10752 [cs]. 1

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted*

intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pages 234–241. Springer, 2015. 3

[29] Omid Saghatchian, Atiyeh Gh. Moghadam, and Ahmad Nickabadi. Cached adaptive token merging: Dynamic token reduction and redundant computation elimination in diffusion model, 2025. 3

[30] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24219–24238, 2022. 5

[31] 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. 2, 3

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 3

[33] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023. 3

[34] Cristian J Vaca-Rubio, Luis Blanco, Roberto Pereira, and Màrius Caus. Kolmogorov-arnold networks (kans) for time series analysis. *arXiv preprint arXiv:2405.08790*, 2024. 3

[35] 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. 3

[36] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 5

[37] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Da-Wei Liu, De mei Li, Hang Zhang, Hao Meng, Hu Wei, Ji-Li Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Min Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiao-Xue Xu, Yi Wang, Yichang Zhang, Yong-An Zhu, Yujia Wu, Yu-Jiao Cai, and Ze-Yang Liu. Qwen-image technical report. *ArXiv*, abs/2508.02324, 2025. 2

[38] 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, 2023. 5

[39] Hanshu Yan, Xingchao Liu, Jiachun Pan, Jun Hao Liew, Qiang Liu, and Jiashi Feng. Perflow: Piecewise rectified flow as universal plug-and-play accelerator. *Advances in Neural Information Processing Systems*, 37:78630–78652, 2024. 3

[40] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan.Zhang, Weihan Wang, Yean Cheng, Bin Xu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[41] Zhihang Yuan, Hanling Zhang, Lu Pu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. DiTFastattn: Attention compression for diffusion transformer models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3

[42] Evelyn Zhang, Bang Xiao, Jiayi Tang, Qianli Ma, Chang Zou, Xuefei Ning, Xuming Hu, and Linfeng Zhang. Token pruning for caching better: 9 times acceleration on stable diffusion for free, 2024. 3

[43] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric, 2018. 5

[44] Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers, 2025. 3

[45] 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 (MM '25)*, page to appear, Dublin, Ireland, 2025. ACM. 3

[46] Haowei Zhu, Dehua Tang, Ji Liu, Mingjie Lu, Jintu Zheng, Jinzhang Peng, Dong Li, Yu Wang, Fan Jiang, Lu Tian, Spandan Tiwari, Ashish Sirasao, Jun-Hai Yong, Bin Wang, and Emad Barsoum. Dip-go: A diffusion pruner via few-step gradient optimization, 2024. 3

[47] Chang Zou, Evelyn Zhang, Runlin Guo, Haohang Xu, Conghui He, Xuming Hu, and Linfeng Zhang. Accelerating diffusion transformers with dual feature caching, 2024. 3

[48] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*. ICLR, 2025. accepted to ICLR 2025. 2, 3

LESA: Learnable Stage-Aware Predictors for Diffusion Model Acceleration

Supplementary Material

6. Method Details

6.1. Preliminary

Diffusion Transformer Architecture. The Diffusion Transformer (DiT) processes an input $\mathbf{x}_t = \{x_i\}_{i=1}^{H\times W}$ through a hierarchical structure $\mathcal{G} = g_1 \circ g_2 \circ \cdots \circ g_L$. Each block g_l, where the superscript l denotes the layer index ranging from 1 to L, is composed as $g_l = \mathcal{F}^l_{\text{SA}} \circ \mathcal{F}^l_{\text{CA}} \circ \mathcal{F}^l_{\text{MLP}}$ and sequentially applies self-attention, cross-attention, and multilayer perceptron components. Critically, these components $\mathcal{F}^l_{\text{SA}}$, $\mathcal{F}^l_{\text{CA}}$, and $\mathcal{F}^l_{\text{MLP}}$ dynamically evolve over time by adapting to the varying noise levels at each denoising timestep. This temporal adaptation allows the model to follow the evolving noise schedule during denoising. Since most adjacent timesteps produce highly similar features, this temporal similarity naturally enables cache-based acceleration.

6.2. Comparison of KAN and MLP

From Kolmogorov–Arnold representation to KAN layers. Kolmogorov–Arnold Network (KAN) is inspired by the classical Kolmogorov–Arnold representation theorem, which states that any continuous multivariate function $f : \mathbb{R}^d \to \mathbb{R}$ can be expressed using compositions and summations of univariate functions. A typical abstract form is

$$f(x_1, \ldots, x_d) = \sum_{q=1}^{Q} \Phi_q \left(\sum_{p=1}^{d} \psi_{q,p}(x_p) \right), \tag{8}$$

where each Φ_q and $\psi_{q,p}$ is a scalar-valued univariate function.

KAN turns this theoretical representation into a trainable neural architecture by parameterizing the involved univariate functions with learnable spline functions. Concretely, a single KAN layer maps $h^{(\ell-1)}(x) \in \mathbb{R}^{D_{\ell-1}}$ to $h^{(\ell)}(x) \in \mathbb{R}^{D_\ell}$ as

$$h_j^{(\ell)}(x) = \sum_{m=1}^{M_\ell} w_{j,m}^{(\ell)} \, \phi_{j,m}^{(\ell)}\big(a_{j,m}^{(\ell)\top} \, h^{(\ell-1)}(x)\big) + b_j^{(\ell)}. \tag{9}$$

Here $j = 1, \ldots, D_\ell$ denotes the output channel index and $a_{j,m}^{(\ell)} \in \mathbb{R}^{D_{\ell-1}}$ are learnable linear projections, $w_{j,m}^{(\ell)} \in \mathbb{R}$ are output mixing weights, and each $\phi_{j,m}^{(\ell)} : \mathbb{R} \to \mathbb{R}$ is a learnable spline-based univariate function. Stacking such layers yields a deep KAN $f(x) = h^{(L)}(x)$ with L KAN layers.

In many applications we are interested in a scalar modulation function $\alpha : \mathbb{R}^d \to \mathbb{R}$, for instance as a temporal weighting term depending on a timestep feature vector $\Delta t \in \mathbb{R}^d$. In this case we can specialize the general layer in Eq. (9) to a *scalar-output* KAN layer. Let $D_0 = d$, $D_1 = 1$, and omit the layer index ℓ and output index j. Then Eq. (9) simplifies to

$$\alpha_{\text{KAN}}(\Delta t) = \sum_{m=1}^{M} w_m \, \phi_m\big(a_m^\top \Delta t\big) + b, \tag{10}$$

where M is the number of spline components, $a_m \in \mathbb{R}^d$ are learnable projection vectors, $w_m \in \mathbb{R}$ are learnable outer weights, and each ϕ_m is a learnable univariate spline.

For notational simplicity, the bias b can be absorbed into an additional constant component. Define $\phi_1(z) \equiv 1$ and $w_1 = b$. Then Eq. (10) can be rewritten as

$$\alpha_{\text{KAN}}(\Delta t) = \sum_{m=1}^{M} w_m \, \phi_m\big(a_m^\top \Delta t\big), \tag{11}$$

Thus, a scalar KAN layer is a linear combination of learnable spline-based responses $\phi_m(a_m^\top \Delta t)$, each acting on a learnable one-dimensional projection of the input.

MLP-based scalar mapping. For comparison, a standard multilayer perceptron (MLP) with one hidden layer and a fixed activation function σ realizes a scalar mapping $\alpha_{\text{MLP}} : \mathbb{R}^d \to \mathbb{R}$ in the form

$$\alpha_{\text{MLP}}(\Delta t) = v^\top \sigma(U \Delta t + c), \tag{12}$$

where $U \in \mathbb{R}^{H\times d}$ and $v \in \mathbb{R}^H$ are learnable weights, $c \in \mathbb{R}^H$ is a bias, and H is the hidden width. Writing out Eq. (12) component-wise gives

$$\alpha_{\text{MLP}}(\Delta t) = \sum_{h=1}^{H} v_h \, \sigma\big(u_h^\top \Delta t + c_h\big), \tag{13}$$

where $u_h^\top$ denotes the h-th row of U.

Formula-level comparison and advantages of KAN. Equations (5) and (12) reveal a common structure:

$$\alpha_{\text{MLP}}(\Delta t) = \sum_{h=1}^{H} v_h \underbrace{\sigma\big(u_h^\top \Delta t + c_h\big)}_{\text{fixed-shape nonlinearity}}, \tag{14}$$

$$\alpha_{\text{KAN}}(\Delta t) = \sum_{m=1}^{M} w_m \underbrace{\phi_m\big(a_m^\top \Delta t\big)}_{\text{learnable spline function}}. \tag{15}$$

The key distinction between MLP and KAN is that MLP uses a fixed activation function for all neurons, whereas KAN **learns the nonlinear functions themselves, making each**

Table 5. **Quantitative comparison in text-to-image generation** for FLUX.1-schnell. Best results are highlighted in **bold**.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
FLUX	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
[schnell]: 4 steps	2.13	1.00×	278.41	1.00×	0.93	34.09	∞	1.00	0.00
FORA ($\mathcal{N}$=3)	1.30	1.64×	139.24	2.00×	**0.95**	34.43	29.77	0.78	0.18
ToCa ($\mathcal{N}$=3,$\mathcal{R}$=80%)	2.20	0.97×	154.54	1.80×	0.94	34.46	29.79	0.78	0.18
DuCa ($\mathcal{N}$=3,$\mathcal{R}$=80%)	1.33	1.60×	145.46	1.91×	**0.95**	34.44	29.80	0.78	0.18
TaylorSeer ($\mathcal{N}$=3,$\mathcal{O}$=2)	**1.29**	**1.65**×	139.24	2.00×	**0.95**	34.43	29.77	0.78	0.18
LESA ($\mathcal{N}$=3)	1.31	1.63×	**139.21**	**2.00**×	**0.95**	**34.48**	**30.31**	**0.80**	**0.16**
FORA ($\mathcal{N}$=4)	1.11	1.92×	69.65	4.00×	0.88	34.27	28.77	0.58	0.45
ToCa ($\mathcal{N}$=4,$\mathcal{R}$=85%)	1.47	1.45×	92.86	3.00×	**0.91**	34.29	28.79	0.62	0.41
DuCa ($\mathcal{N}$=4,$\mathcal{R}$=85%)	**1.03**	**2.07**×	88.31	3.15×	0.90	34.30	28.78	0.61	0.42
TaylorSeer ($\mathcal{N}$=4,$\mathcal{O}$=2)	1.11	1.92×	69.65	4.00×	0.88	34.27	28.77	0.58	0.45
LESA ($\mathcal{N}$=4)	1.15	1.85×	**69.61**	**4.00**×	**0.91**	**34.67**	**29.21**	**0.69**	**0.30**

parameter far more expressive. In MLP, all hidden units rely on the same fixed activation function σ (e.g., SiLU), which limits nonlinear behavior to a predetermined template and requires additional depth or width to increase expressiveness. In contrast, KAN assigns a learnable spline function ϕ_m to each component, allowing the nonlinear response itself to adapt its shape to data. This flexibility enables KAN to represent complex or highly non-uniform scalar dynamics with far fewer components than an MLP, leading to substantially improved parameter efficiency.

6.3. Scalar Modulation

Taking the 2nd-order Taylor predictor (e.g., TaylorSeer) as a reference, given history features $\{h_3, h_2, h_1\}$ at timestamps $\{t_3, t_2, t_1\}$, the finite difference formulation is:

$$\delta h_2 = \frac{h_2 - h_3}{t_2 - t_3}, \quad \delta h_1 = \frac{h_1 - h_2}{t_1 - t_2},$$
$$\delta^2 h_1 = \frac{\delta h_1 - \delta h_2}{t_1 - t_2}, \tag{16}$$
$$h_0 = h_1 + \delta h_1 \cdot (t_0 - t_1) + \frac{\delta^2 h_1}{2} \cdot (t_0 - t_1)^2.$$

By expanding and rearranging the terms for h_0, we can express $h_0 = C_1 h_1 + C_2 h_2 + C_3 h_3$, where C_1, C_2, C_3 are polynomial functions of the timesteps. LESA adopts fused history features ($Z \approx \sum h_i$) and a scalar modulation ($\alpha \approx \sum C_i$). We can expand the product $\alpha Z = (\sum C_i)(\sum h_i)$:

$$\begin{aligned} \alpha Z = & \underbrace{C_1 h_1 + C_2 h_2 + C_3 h_3}_{\text{Main Terms}} \\ & + \underbrace{C_1 h_2 + C_1 h_3 + C_2 h_1 + C_2 h_3 + C_3 h_1 + C_3 h_2}_{\text{Cross Terms}} \end{aligned} \tag{17}$$

This reveals that LESA covers the "Main Terms" of standard Taylor expansion while introducing learnable "Cross Terms." This gives the model **a broader learning space** to capture feature interactions beyond fixed derivative rules.

7. Experiments Details

7.1. Model Configuration & Evaluation and Metrics

We evaluate our proposed method on five representative large-scale diffusion models: FLUX.1-dev[11], FLUX.1-schnell[11], Qwen-Image[37], Qwen-Image-Lightning[37], and HunyuanVideo [10].

FLUX.1-dev and FLUX.1-schnell For both FLUX.1-dev and FLUX.1-schnell, image generation is performed at a resolution of 1024×1024 using 200 prompts sampled from the DrawBench benchmark. Model performance is evaluated with ImageReward, CLIP Score, PSNR, SSIM, and LPIPS, covering both text–image alignment and low-level visual fidelity. All latency results are obtained on an A100 GPU, with identical inference configurations to ensure fair comparison.

Qwen-Image and Qwen-Image-Lightning For Qwen-Image and Qwen-Image-Lightning, image generation is conducted on the same DrawBench-200 prompt set for consistent comparison. Images are produced at a resolution of 1328×1328, and the same metrics ImageReward, CLIP Score, PSNR, SSIM, and LPIPS are used to assess both text–image alignment and overall visual quality. Inference latency for Qwen-Image is recorded on an H20 GPU, while Qwen-Image-Lightning is evaluated on an A100 GPU, each under identical sampling and batch settings.

Stable Diffusion XL For Stable Diffusion XL, experiment is performed at a resolution of 1024×1024 using the same 200 prompts from the DrawBench benchmark as in the previous experiments. We adopt the same evaluation protocol to jointly measure text–image alignment and low-level visual fidelity. All latency results are obtained on a single A100 GPU with identical sampling configurations and batch settings for both LESA and DeepCache to ensure a fair comparison.

HunyuanVideo For HunyuanVideo, we adopt a video generation configuration of 480×640 spatial resolution, 65 frames, and 50 inference steps. Both the baseline model and

Table 6. **Quantitative comparison in text-to-image generation** for Qwen-Image-Lightning. Best results are highlighted in **bold**.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
Qwen	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
[Lightning]: 8 steps	4.37	1.00×	560.96	1.00×	1.29	35.32	∞	1.00	0.00
FORA ($\mathcal{N}$=2)	3.08	1.42×	350.65	1.60×	1.27	35.20	31.35	0.82	0.13
ToCa ($\mathcal{N}$=2, $\mathcal{R}$=90%)	6.70	0.65×	393.19	1.43×	1.27	35.30	32.03	0.81	0.15
DuCa ($\mathcal{N}$=2, $\mathcal{R}$=90%)	3.37	1.30×	379.26	1.48×	**1.28**	35.16	32.74	0.84	0.11
TaylorSeer ($\mathcal{N}$=2)	3.23	1.35×	350.65	1.60×	1.27	**35.45**	29.94	0.68	0.28
LESA ($\mathcal{N}$=2)	**2.95**	**1.48**×	**350.61**	**1.60**×	**1.28**	35.20	**33.42**	**0.85**	**0.10**
FORA ($\mathcal{N}$=3)	2.63	1.66×	280.55	2.00×	1.26	35.10	29.70	0.75	0.20
ToCa ($\mathcal{N}$=3, $\mathcal{R}$=95%)	5.72	0.76×	318.50	1.76×	1.26	35.13	30.40	0.75	0.21
DuCa ($\mathcal{N}$=3, $\mathcal{R}$=95%)	2.79	1.57×	292.46	1.92×	**1.27**	35.11	30.25	0.76	0.19
TaylorSeer ($\mathcal{N}$=3)	2.78	1.57×	280.55	2.00×	1.24	**35.48**	28.91	0.63	0.35
LESA ($\mathcal{N}$=3)	**2.48**	**1.76**×	**280.49**	**2.00**×	**1.27**	35.24	**31.38**	**0.80**	**0.16**
FORA ($\mathcal{N}$=4)	1.73	2.53×	140.34	3.99×	1.19	35.86	28.41	0.53	0.48
ToCa ($\mathcal{N}$=4, $\mathcal{R}$=95%)	3.61	1.21×	197.75	2.84×	1.16	35.95	28.75	0.54	0.52
DuCa ($\mathcal{N}$=4, $\mathcal{R}$=95%)	2.06	2.12×	161.79	3.47×	1.22	**36.10**	28.75	0.53	0.49
TaylorSeer ($\mathcal{N}$=4)	1.79	2.44×	140.34	3.99×	1.22	35.81	28.48	0.57	0.42
LESA ($\mathcal{N}$=4)	**1.52**	**2.88**×	**140.25**	**4.00**×	**1.23**	35.66	**28.85**	**0.63**	**0.37**

our caching-accelerated variant are evaluated using three standard video quality metrics: PSNR, SSIM, and LPIPS. To reduce sample-wise variance, we utilize 956 prompts, each generating one video, ensuring statistically stable results across diverse scenarios. All latency measurements for video experiments are performed on an H800 GPU.

7.2. Training Details

During training, we use a custom set of 100 prompts and first run one epoch of Ground-Truth Guided Training. We then load the weights from this stage and perform two additional epochs of Closed-Loop Autoregressive Training. The KAN predictor is configured with a hidden dimension of 256 and is implemented in bfloat16 on GPU when available. We optimize the model with AdamW using a learning rate of 1×10^{-4}, weight decay of 1×10^{-4}, together with gradient clipping at a maximum norm of 1.0.

7.3. More Experimental Results

7.3.1. FLUX.1-schnell

As shown in Tab. 5, *LESA* ($\mathcal{N}$=4) reduces FLOPs from 278.41T to 69.61T, achieving a 4× speedup and lowering latency from 2.13 s to 1.15 s. Despite this remarkable acceleration, LESA maintains the best visual quality, achieving the highest ImageReward, CLIP score, PSNR, SSIM, and LPIPS compared with FORA, TaylorSeer, ToCa, and DuCa.

7.3.2. Qwen-Image-Lightning

As shown in Tab. 6, *LESA* consistently offers a better speed–quality trade-off than cache-based baselines across all acceleration levels ($\mathcal{N}$=2 ∼ 4). Focusing on $\mathcal{N}$=3, *LESA* attains a 1.76× latency speedup and 2.00× FLOP reduction over the 8-step baseline. At this operating point, LESA yields higher visual quality, with PSNR of 31.38, SSIM of 0.80, and LPIPS of 0.16, versus 29.70, 0.75, and 0.20 for FORA and 28.91, 0.63, and 0.35 for TaylorSeer.

Table 7. **Quantitative comparison of *LESA* and DeepCache in text-to-image generation** for Stable Diffusion XL.

Method	Quality Metrics		Perceptual Metrics		
Stable Diffusion XL	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
[xl]: 50 steps	0.58	33.63	∞	1.00	0.00
DeepCache ($\mathcal{N}$=3)	0.52	33.42	29.85	0.77	0.26
LESA ($\mathcal{N}$=3)	0.56	33.57	31.28	0.80	0.18
DeepCache ($\mathcal{N}$=4)	0.40	33.29	29.22	0.73	0.31
LESA ($\mathcal{N}$=4)	0.52	33.39	30.47	0.75	0.24
DeepCache ($\mathcal{N}$=5)	0.40	33.07	28.85	0.70	0.36
LESA ($\mathcal{N}$=5)	0.41	33.36	30.02	0.73	0.28
DeepCache ($\mathcal{N}$=6)	0.34	32.87	28.66	0.69	0.39
LESA ($\mathcal{N}$=6)	0.41	33.33	29.69	0.70	0.31
DeepCache ($\mathcal{N}$=7)	0.28	32.56	28.51	0.66	0.43
LESA ($\mathcal{N}$=7)	0.37	33.20	29.45	0.69	0.34
DeepCache ($\mathcal{N}$=8)	0.19	32.27	28.43	0.66	0.44
LESA ($\mathcal{N}$=8)	0.33	32.97	29.23	0.67	0.38
DeepCache ($\mathcal{N}$=9)	0.12	32.20	28.35	0.64	0.47
LESA ($\mathcal{N}$=9)	0.21	32.48	29.09	0.67	0.40
DeepCache ($\mathcal{N}$=10)	-0.01	31.81	28.29	0.63	0.51
LESA ($\mathcal{N}$=10)	0.22	32.56	28.94	0.65	0.43

7.3.3. Stable Diffusion XL

On Stable Diffusion XL, as shown in Tab. 7, *LESA* consistently delivers better visual quality than DeepCache across all acceleration levels $\mathcal{N}$ from 3 to 10, with higher ImageReward and CLIP scores, higher PSNR and SSIM, and lower LPIPS. Compared to DeepCache at $\mathcal{N}$=10, *LESA* increases ImageReward from -0.01 to 0.22 and CLIP score from 31.81 to 32.56, while PSNR rises from 28.29dB to 28.94dB and LPIPS drops from 0.51 to 0.43.

Table 8. **Comparison of methods in Cache Memory, MACs, Latency, and FLOPS on FLUX-1.dev**. Best results are highlighted in **bold**.

Method	VRAM (GB)↓	MACs (T)↓	Latency (s)↓	FLOPs (T)↓	Image Reward↑	PSNR↑
[dev]: 50 steps	0.62	1859.62	23.24	3726.87	0.99	∞
ToCa($\mathcal{N}$=8, $\mathcal{R}$=75%)	12.31	414.88	12.39	829.86	0.95	29.07
DuCa($\mathcal{N}$=8, $\mathcal{R}$=70%)	3.65	428.86	9.40	858.27	0.94	29.06
TeaCache(l=1.0)	**0.69**	409.43	7.07	820.55	0.84	28.61
TaylorSeer($\mathcal{N}$=6, O=2)	7.66	372.38	6.73	746.28	**1.02**	28.94
LESA($\mathcal{N}$=7)	0.81	**372.25**	**5.19**	**745.51**	0.98	**30.17**

- **Note:** All methods inherit baseline memory optimizations (e.g., FlashAttention). ToCa is incompatible with these, hence its higher reported cache usage. **Actual cache memory usage** for each method should be computed as: `Peak VRAM during inference − Baseline VRAM before inference`.

Table 9. **Quantitative comparison of *LESA* trained with different number of prompts** for QwenImage. Best results are highlighted in **bold**.

Method	Quality Metrics		Perceptual Metrics		
QwenImage	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
50 steps	1.25	35.59	∞	1.00	0.00
LESA (1 prompt)	1.14	35.17	30.18	**0.78**	0.25
LESA (5 prompts)	1.15	35.31	**30.36**	**0.78**	0.25
LESA (10 prompts)	**1.16**	35.33	30.21	**0.78**	0.26
LESA (20 prompts)	1.15	35.21	29.78	0.77	0.25
LESA (50 prompts)	1.15	35.32	29.96	**0.78**	**0.24**
LESA (100 prompts)	1.15	**35.36**	30.18	**0.78**	0.25

7.3.4. More Ablation Studies

Impact of number of prompts on training performance As shown in Tab. 9, at $\mathcal{N}$=7, increasing the number of prompts for LESA on QwenImage yields only marginal improvements. Using 5 prompts gives the highest PSNR (30.36), 10 prompts achieves the best ImageReward (1.16), and 100 prompts obtains the highest CLIP score (35.36). Overall, the metrics vary only slightly across different prompt counts, indicating that LESA already delivers strong performance with as few as 5–10 prompts.

Memory usage comparison of different caching methods As shown in Tab. 8, TeaCache achieves the lowest VRAM usage (0.69 GB), while LESA keeps VRAM low (0.81 GB) and attains the best computational efficiency, with the fewest MACs (372.25 T), lowest latency (5.19 s), and lowest FLOPs (745.51 T). TaylorSeer slightly outperforms others in Image Reward (1.02), whereas LESA achieves the highest PSNR (30.17), indicating better reconstruction quality. Overall, LESA providprovides the best trade-off between memory, efficiency, and image quality among the compared cache methods.