LinCa: Accelerating Diffusion Models via Learnable Decomposed Feature Caching

Jinshan Liu[1,3*], **Haoran Qin**[1,4*], **Xiaobing Tu**[2], **Jiacheng Liu**[1], **Jiahui Hu**[1,5], **Zhengan Yan**[1], **Yukun Xie**[1,3], **Kerui Shen**[1,3], **Jinkui Ren**[2], **Yuqi Lin**[1,6], **Xiantao Zhang**[2], and **Linfeng Zhang**[1†]

[1]Shanghai Jiao Tong University, China [2]Terminal Intelligent Computing Division, Alibaba Cloud, China [3]Xi'an Jiaotong University, China [4]Shandong University, China [5]South China University of Technology, China [6]Jilin University, China

Fig. 1: Images sampled by Qwen-Image with *LinCa* at **6.95**× acceleration.

Abstract. Diffusion models have achieved remarkable success in image and video generation, yet the high computational cost of iterative sampling remains a critical bottleneck for practical deployment. Feature caching has emerged as a promising acceleration paradigm by reusing or predicting intermediate features across timesteps. However, existing training-free methods apply uniform prediction strategies that cannot adapt to the heterogeneous feature dynamics, causing significant quality degradation under high acceleration ratios. We propose **LinCa**, a feature caching framework based on learnable invertible networks. LinCa decomposes cached features into sub-components with distinct continuity properties via a lightweight invertible network and applies differentiated prediction orders matched to each component. The strict invertibility guarantees lossless reconstruction back to the original feature space, forming a unified Decompose-Predict-Reconstruct pipeline. By training separate predictors for different models and timestep segments, LinCa adapts to heterogeneous feature dynamics. Experiments on FLUX, Qwen-Image, and HunyuanVideo demonstrate that LinCa, with less than 0.2% additional parameters, significantly outperforms existing methods and maintains near-lossless quality at 5-7× speedup. Code: https://github.com/QHR69/LinCa.

Keywords: Diffusion Acceleration · Learnable Invertible Network

* Equal contribution.
† Corresponding author. Email: zhanglinfeng@sjtu.edu.cn

1 Introduction

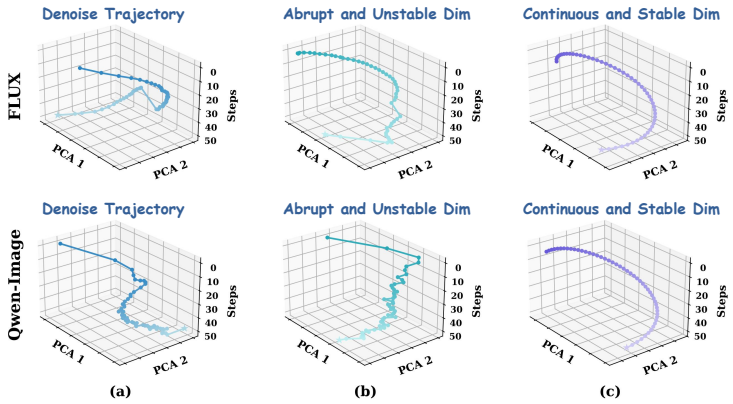

Fig. 2: Analysis of dynamics mismatch. (a): Denoise trajectories visualized via Principal Component Analysis (PCA) within FLUX.1-dev and Qwen-Image. Significant continuity differences exist across different models and denoising stages. **(b)-(c):** Denoise trajectories of specific dimensions visualized via PCA within FLUX.1-dev and Qwen-Image. Even within the same model and stage, significant continuity differences still exist across different feature dimensions. (b) shows dimensions with low continuity and abrupt mutations, while (c) shows dimensions with high continuity and stability.

Diffusion models have achieved remarkable progress in image and video generation and editing [2,15,38]. Recently, Diffusion Transformers [37] have become the predominant architecture for high-quality conditional generation and editing, owing to their superior scalability and modeling capacity. However, the iterative denoising sampling mechanism of Diffusion Transformers requires multi-step forward passes, and the resulting high computational cost makes efficient inference a critical bottleneck for practical deployment.

To address this challenge, two main acceleration directions have emerged: reducing the total number of sampling steps through algorithmic advances [29], and lowering the per-step cost through architectural optimization [51,54]. Among these, feature caching exploits the temporal consistency of hidden representations across adjacent timesteps and has become a particularly promising solution. Specifically, it performs a full model forward pass only once every N timesteps and caches the intermediate features. The remaining $N-1$ steps directly reuse or predict features based on previous caches, thereby skipping expensive forward computation to achieve acceleration, as exemplified by recent works such as FORA, ToCa, and TaylorSeer [25, 41, 60]. Despite this progress, we identify the following limitations in current feature caching methods.

Cross-timestep and cross-model dynamics mismatch. Existing feature caching methods implicitly assume that the hidden features of diffusion models follow a uniform evolution pattern throughout the entire denoising process, and thus apply a single preset caching strategy across all timesteps and all models. However, as shown in Figure 2(a), we find that even within the same model, the

feature evolution patterns differ significantly across timesteps. Some timestep segments exhibit stable and continuous trajectories, while others display poor continuity accompanied by abrupt mutations. Across different models, the discrepancy in feature trajectory shapes and variation patterns is more pronounced. This indicates that the denoising process is not governed by a single dynamics mechanism, and applying a uniform caching rule across all timesteps and models inevitably introduces cumulative errors in certain scenarios.

Cross-dimension dynamics mismatch. From a finer-grained perspective, existing methods apply a uniform prediction strategy to all feature dimensions, implicitly assuming that all dimensions share the same continuity. However, as shown in Figure 2(b)-(c), even within the same model at the same timestep segment, different feature dimensions exhibit markedly different dynamics. Some dimensions display good stability and continuity, making them suitable for high-order polynomial prediction. Other dimensions are unstable and accompanied by abrupt mutations, making them difficult to predict directly. More critically, dimensions with different continuity properties are interleaved in the original feature space and cannot be separated by simple dimensional partitioning. This suggests that a uniform prediction strategy cannot accommodate the vastly different dynamics across feature dimensions.

To address the dynamics mismatch at both levels, this paper introduces **LinCa**, a **L**earnable **I**nvertible **N**etworks based Feature **Ca**ching acceleration framework. Through minimal parameter overhead and training cost, LinCa employs a learnable mapping to decompose cached features into sub-components with different dynamics and applying the corresponding differentiated prediction orders to each component. Unstable components are directly reused from the nearest cache, while components with good continuity are predicted via polynomial extrapolation matched to their continuity from historical caches. This learnable mapping is realized by a lightweight fully invertible network whose strict invertibility guarantees that the decomposed features can be losslessly reconstructed back to the original feature space, forming an end-to-end learnable Decompose-Predict-Reconstruct pipeline. Furthermore, we partition the denoising timesteps into multiple segments and train isomorphic but independently parameterized predictors, adapting to the distinct feature dynamics of different diffusion models and timestep segments. With a parameter count less than 0.2% of the original diffusion model, LinCa requires only 100-200 pre-generated features as training data, and completes training within 1 hour on a single GPU (12GB VRAM) without loading the diffusion model weights. During inference, LinCa achieves nearly the same speed as training-free methods, further enhancing its practical value.

LinCa delivers robust and efficient feature prediction across diverse tasks and architectures. It achieves near-lossless acceleration of **5.51**× on FLUX.1-dev, **6.95**× on Qwen-Image, **7.08**× on Qwen-Image-Edit, and **5.50**× on HunyuanVideo (Figure 1). Moreover, it is also fully compatible with distillation or quantization with higher acceleration ratio and strong image quality. In summary, our main contributions are as follows:

- **Heterogeneous Feature Dynamics:** We reveal that the hidden features of diffusion models exhibit significant dynamics mismatch across timesteps and models, and that different feature dimensions display markedly different evolution patterns, highlighting the importance of adaptively matching feature dynamics through learnable mappings.
- **LinCa Framework:** We propose **LinCa**, a feature caching framework that decomposes cached features via a lightweight invertible network and applies differentiated prediction orders. By training separately for different models across different timestep segments, LinCa adapts to heterogeneous feature dynamics with minimal training overhead.
- **Superior Performance:** We evaluate LinCa on diverse architectures and tasks including FLUX.1-dev, Qwen-Image, Qwen-Image-Edit, HunyuanVideo and even distilled or quantized models. Across all settings, LinCa significantly outperforms existing training-free methods at the same acceleration ratio and maintains near-lossless generation quality under high speedup.

2 Related Works

Diffusion models [16, 44] have become the dominant framework for high-fidelity image and video generation. The Diffusion Transformer (DiT) [37] further advanced this field through superior scalability and expressive capacity, becoming foundational for large-scale visual generation systems [5, 6, 31–35, 50, 57]. Nevertheless, the iterative nature of the sampling process remains the primary inference bottleneck, and current acceleration efforts focus on two complementary directions: reducing sampling steps and accelerating the denoising network.

2.1 Sampling Timestep Reduction

DDIM [45] introduced deterministic few-step sampling, further refined by the DPM-Solver series [29, 30] through high-order ODE solvers. Rectified Flow [28] shortens the transport path via optimal transport, while knowledge distillation [36, 40] compresses long sampling trajectories into compact student generators. Consistency Models [46] further enable single-step synthesis by learning a direct noise-to-data mapping. Although effective, these methods typically require redesigning sampling algorithms or retraining the diffusion model, limiting their applicability to pre-trained diffusion models.

2.2 Denoising Network Acceleration

Another acceleration direction focuses on reducing the computational cost of each forward pass, primarily through model compression and feature caching.

Model Compression-based Acceleration. Model compression reduces inference overhead through structured pruning [11, 59], quantization [19, 42], distillation [21], and token merging or pruning [3, 8, 18, 52, 53]. However, these methods often necessitate additional training or fine-tuning to maintain generation quality, and aggressive compression may compromise robustness [21].

Feature Caching-based Acceleration. Feature caching avoids redundant computation by reusing activations across timesteps, attracting wide attention for its ability to work without modifying model structure and typically in a training-free manner. Current works [4, 24] extended caching to DiT architectures: FORA [41] and Δ-DiT [7] cache block outputs, while ToCa and DuCa [60, 61] adaptively reuse tokens or regions. TaylorSeer [25] introduced polynomial extrapolation from cached features, further advanced by FoCa [56] and SpeCa [26]. More recently, Clusca [58] reduces token redundancy via spatial clustering, while HyCa [55] applies dimension-wise caching based on mixed ODE modeling.

However, all the above methods treat cached features as a whole, applying a uniform prediction strategy across all feature dimensions and overlooking their markedly different continuity. Meanwhile, feature dynamics differ significantly across model architectures and denoising stages, yet existing training-free methods apply the same strategy across all scenarios, leading to cumulative prediction errors and notable quality degradation under high acceleration ratios.

In contrast, **LinCa** employs invertible networks capable of lossless reconstruction to map cached features into sub-components with distinct continuity properties for differentiated order prediction, addressing the quality degradation inherent in existing caching methods. Meanwhile, data-driven per-segment training enables LinCa to adaptively accommodate the feature dynamics differences across model architectures and denoising stages with minimal training cost, significantly improving prediction performance under high acceleration ratios.

3 Methodology

3.1 Preliminary

Feature caching accelerates diffusion model inference by avoiding redundant computation across timesteps. Let $\mathbf{x}_t$ denote the hidden feature produced by the diffusion model at timestep t, feature caching perform a full forward pass every N steps and cache the intermediate features, while estimating the remaining $N-1$ steps through a prediction function f based on historical caches:

$$\hat{\mathbf{x}}_{t-k} = f(\mathbf{x}_t, \mathbf{x}_{t+N}, \ldots), \quad k \in \{1, \ldots, N-1\} \tag{1}$$

where $\mathbf{x}_t, \mathbf{x}_{t+N}, \ldots$ are previously cached features. Existing feature caching methods aim to construct an effective f for accurate prediction. For instance, when f is modeled as the identity mapping, it corresponds to direct reuse: $\hat{\mathbf{x}}_{t-k} = \mathbf{x}_t$. Alternatively, f can be modeled as a Taylor expansion: $\hat{\mathbf{x}}_{t-k} = \mathbf{x}_t + \sum_{i=1}^{m} \frac{\Delta^i \mathbf{x}_t}{i! \cdot N^i} (-k)^i$, where m is the prediction order and $\Delta^i \mathbf{x}_t$ denotes the i^{th} order discrete difference. f can also be modeled as Hermite interpolation based on discrete differences: $\hat{\mathbf{x}}_{t-k} = \mathbf{x}_t + \sum_{i=1}^{m} \alpha_i(k)\, \Delta^i \mathbf{x}_t$, where $\alpha_i(k)$ are interpolation coefficients derived from Hermite polynomials.

However, regardless of the form of f, existing methods apply a uniform prediction strategy and prediction order across all models, all timesteps, and all feature dimensions, failing to accommodate the significant dynamics differences at each of these levels and thus compromising generation quality.

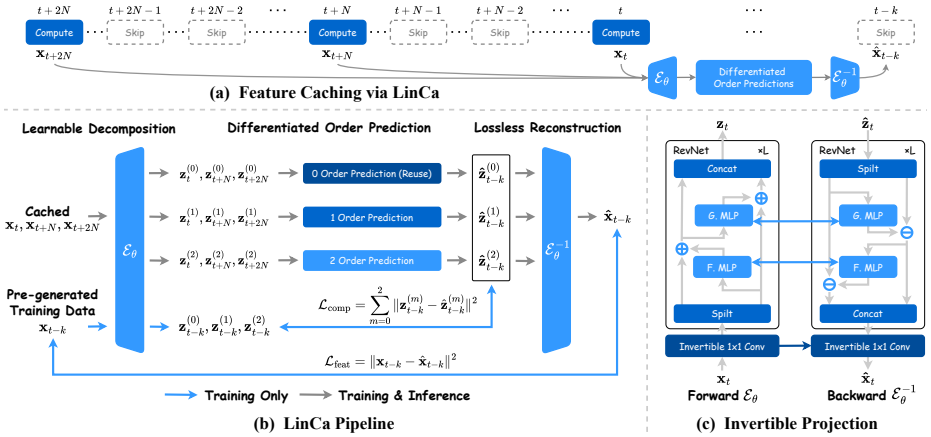

Fig. 3: Overview of the LinCa framework. (a) Feature Caching via LinCa: Feature caching caches features during computation steps and skips computation by utilizing historical cached features during prediction steps, where LinCa provides an end-to-end "Decompose-Predict-Reconstruct" pipeline. **(b) LinCa Pipeline:** Cached features are decomposed into sub-components with distinct continuity, predicted by differentiated order polynomial extrapolation, and losslessly reconstructed back to the original feature space. **(c) Invertible Projection:** Each invertible block consists of an invertible 1×1 convolution and an additive coupling layer with lightweight MLPs F and G. The strict invertibility of each sub-layer guarantees lossless reconstruction.

3.2 Learnable Invertible Network based Feature Caching

In this section, we introduce the **LinCa** framework (Figure 3). To address the dynamics mismatch at multiple levels in existing methods, LinCa is built upon two key components: (i) decomposing cached features into sub-components with distinct continuity properties through a learnable mapping and applying order-matched polynomial prediction to each sub-component, and (ii) realizing this mapping with a lightweight invertible network whose strict invertibility guarantees lossless reconstruction of the input features, trained separately for different models and timestep segments to adapt to their distinct feature dynamics.

Learnable Decomposition and Differentiated Order Prediction. Our differentiated prediction strategy is motivated by the observation that different feature dimensions exhibit distinct dynamics. As shown in Figure 2(b)-(c), some feature dimensions are unstable with abrupt mutations and low temporal continuity, making them difficult to predict. In contrast, other dimensions exhibit varying degrees of temporal continuity, with diffusion trajectories suitable for polynomial prediction at different orders. However, dimensions with different continuity properties are interleaved in the original feature space and cannot be separated by simple partitioning. We therefore propose to learn a differentiated decomposition strategy that regroups and separates feature dimensions with different evolution characteristics, applying prediction functions f of different orders to each group.

To this end, following prior work [23], we cache the cumulative residual feature at the final layer of the diffusion model. For the cached feature $\mathbf{x}_t \in \mathbb{R}^{N\times D}$ at timestep t, LinCa projects it through a learnable mapping $\mathcal{E}_\theta$ and decomposes it along the feature dimension into M sub-components ($M = 3$ in practice):

$$\mathbf{z}_t = \mathcal{E}_\theta(\mathbf{x}_t) = [\mathbf{z}_t^{(0)} \mid \mathbf{z}_t^{(1)} \mid \cdots \mid \mathbf{z}_t^{(M-1)}], \quad \mathbf{z}_t^{(m)} \in \mathbb{R}^{N\times d_m} \tag{2}$$

where $\mathcal{E}_\theta$ is continuously optimized during training to cluster dimensions with similar continuity into corresponding sub-spaces, thereby regrouping and separating feature dimensions with different evolution characteristics. Based on the distinct dynamics of each sub-component, we apply differentiated order prediction strategies. For the 0^{th} order sub-component, which is unstable with weak continuity, we directly reuse the nearest cached value: $\hat{\mathbf{z}}_t^{(0)} = \mathbf{z}_{t_{\text{prev}}}^{(0)}$. For higher-order sub-components ($m \geq 1$), we predict each using m^{th} order Hermite interpolation. These closed-form predictors accurately reconstruct higher-order details at negligible computational cost. Finally, all sub-components are concatenated and reconstructed back to the original feature space through the inverse mapping $\mathcal{E}_\theta^{-1}$, yielding the final predicted feature:

$$\hat{\mathbf{x}}_t = \mathcal{E}_\theta^{-1}([\hat{\mathbf{z}}_t^{(0)} \mid \hat{\mathbf{z}}_t^{(1)} \mid \cdots \mid \hat{\mathbf{z}}_t^{(M-1)}]) \tag{3}$$

Lossless Mapping via Invertible Networks. The above Decompose-Predict-Reconstruct pipeline requires $\mathcal{E}_\theta$ to be both learnable and capable of losslessly reconstructing the predicted sub-components back to the original feature space through its inverse $\mathcal{E}_\theta^{-1}$. To achieve this, following [13], we design a lightweight invertible network whose architecture guarantees strict mathematical invertibility, ensuring that the reconstruction process introduces no additional loss.

Concretely, $\mathcal{E}_\theta$ consists of L stacked invertible blocks. Each block contains two invertible sub-layers: an invertible 1×1 convolution (parameterized by an orthogonal matrix $\mathbf{W}$) for channel mixing, followed by an additive coupling layer that evenly splits the features along the feature dimension into $\mathbf{u}_1, \mathbf{u}_2$ and applies lightweight MLPs F, G:

$$\mathbf{v}_1 = \mathbf{u}_1 + F(\mathbf{u}_2), \quad \mathbf{v}_2 = \mathbf{u}_2 + G(\mathbf{v}_1) \tag{4}$$

The outputs $\mathbf{v}_1, \mathbf{v}_2$ are concatenated along the feature dimension. During reconstruction, the corresponding inverse mapping is executed: first recovering the pre-coupling components via subtraction $\mathbf{u}_2 = \mathbf{v}_2 - G(\mathbf{v}_1)$, $\mathbf{u}_1 = \mathbf{v}_1 - F(\mathbf{u}_2)$, then restoring the channel mixing through $\mathbf{W}^{-1}$. Since each sub-layer is strictly invertible, the entire network satisfies $\mathcal{E}_\theta^{-1} \circ \mathcal{E}_\theta = \mathbf{I}$, introducing no additional error and thus guaranteeing the highest reconstruction quality.

To adapt to the feature dynamics of different timestep segments, we partition the denoising process into S segments and independently train an isomorphic but separately parameterized predictor $\mathcal{E}_\theta^{(s)}$ for each segment. LinCa requires only 100-200 pre-generated image features and completes training within one hour

on a single GPU (12GB VRAM) without loading the diffusion model weights, adapting to the feature dynamics of different diffusion models at minimal cost.

In detail, during training, we load the target feature $\mathbf{x}_{t-k}$ and historical cached features $\mathbf{x}_t, \mathbf{x}_{t+N}, \ldots$ from pre-generated data, and decompose them through $\mathcal{E}_\theta^{(s)}$ into sub-components $\mathbf{z}_{t-k}^{(m)}$ and $\mathbf{z}_t^{(m)}, \mathbf{z}_{t+N}^{(m)}, \ldots$ respectively. The historical sub-components are then used to predict the target sub-components $\hat{\mathbf{z}}_{t-k}^{(m)}$ via differentiated order extrapolation, and the predictions are reconstructed through the inverse mapping $(\mathcal{E}_\theta^{(s)})^{-1}$ to yield $\hat{\mathbf{x}}_{t-k}$. The optimization objective for each segment is:

$$\mathcal{L}^{(s)} = \mathcal{L}_{\text{feat}}^{(s)} + \lambda \mathcal{L}_{\text{comp}}^{(s)} \tag{5}$$

where the end-to-end prediction loss $\mathcal{L}_{\text{feat}}^{(s)} = \|\hat{\mathbf{x}}_{t-k} - \mathbf{x}_{t-k}\|^2$ ensures the overall prediction quality in the original feature space after inverse reconstruction. The sub-component prediction loss $\mathcal{L}_{\text{comp}}^{(s)} = \sum_{m=0}^{M-1} \|\hat{\mathbf{z}}_{t-k}^{(m)} - \mathbf{z}_{t-k}^{(m)}\|^2$ directly constrains the prediction accuracy of each sub-component, driving $\mathcal{E}_\theta^{(s)}$ to cluster dimensions suitable for m^{th} order prediction into the corresponding sub-space.

4 Experiments

4.1 Experiment Settings

Model Configurations. The experiments are conducted on four state-of-the-art diffusion-based models: the text-to-image models FLUX.1-dev [20] and Qwen-Image [48], the text-to-video model HunyuanVideo [47], and the image editing model Qwen-Image-Edit [48]. To further assess compatibility with model compression techniques, we also evaluate our method on distilled or quantized models: FLUX.1-lite-8B [12], FLUX.1-schnell [1] and FLUX.1-dev-int8 [10]. All experiments are conducted on NVIDIA A100 GPUs for the FLUX series, H100 GPUs for HunyuanVideo, and H20 GPUs for Qwen-Image and Qwen-Image-Edit. More details are provided in the supplementary materials.

Evaluation and Metrics. For text-to-image generation, we follow the DrawBench [39] protocol and evaluate all models on a fixed set of 200 prompts, assessing images using ImageReward [49] for photorealism, CLIP Score [14] for text-image alignment, and PSNR, SSIM, and LPIPS for fidelity. For text-to-video generation, we evaluate our model on VBench [17], which provides multi-dimensional assessments covering motion quality, visual appearance, and semantic consistency. Regarding image editing tasks, we utilize GEdit-Bench [27] to evaluate model performance across a diverse set of edit types and prompts.

4.2 Results on Text-to-Image Generation

As shown in Table 1, LinCa achieves the best speed-quality trade-off on **FLUX.1-dev**. At $N = 4$, it reaches an ImageReward of **1.0175** with **3.32**× acceleration, surpassing FoCa (0.9917 at 2.80×) and TaylorSeer (1.0018 at 2.82×). At $N = 6$,

Table 1: Quantitative comparison of text-to-image generation for FLUX.1-dev. Best results are highlighted in **bold**, and second-best are underlined.

Method	Acceleration				ImageReward ↑	CLIP Score ↑
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
Original: 50 steps	23.10	1.00×	3719.50	1.00×	0.9930 (+0.0%)	32.61 (+0.0%)
60% steps	14.87	1.55×	2231.70	1.67×	0.9693 (−2.4%)	32.50 (−0.3%)
Δ−DiT ($\mathcal{N}=2$) [7]	15.96	1.45×	2480.01	1.50×	0.9471 (−4.6%)	32.46 (−0.5%)
Δ−DiT ($\mathcal{N}=3$) [7]	11.63	1.98×	1686.76	2.21×	0.8750 (−11.9%)	32.29 (−1.0%)
FORA ($\mathcal{N}=3$) [41]	9.08	2.54×	1320.07	2.82×	0.9802 (−1.3%)	32.45 (−0.5%)
DBCache ($\mathcal{F}=8, B=8$) [9]	15.05	1.53×	2384.29	1.56×	1.0097 (+1.7%)	32.72 (+0.3%)
TaylorSeer ($\mathcal{N}=3, O=2$) [25]	8.83	2.61×	1320.07	2.82×	1.0018 (+0.9%)	32.58 (−0.1%)
FoCa ($\mathcal{N}=3$) [56]	8.35	2.78×	1327.21	2.80×	0.9917 (−0.1%)	32.75 (+0.4%)
LinCa ($\mathcal{N}=4$)	**7.51**	**3.08×**	**1120.68**	**3.32×**	**1.0175** (+2.5%)	**32.88** (+0.8%)
34% steps	8.10	2.85×	1264.63	3.13×	0.9482 (−4.5%)	32.28 (−1.0%)
Chipmunk [43]	11.39	2.02×	1505.87	2.47×	0.9965 (+0.4%)	32.71 (+0.3%)
FORA ($\mathcal{N}=4$) [41]	7.28	3.14×	967.91	3.84×	0.9757 (−1.7%)	32.31 (−0.9%)
`ToCa` ($\mathcal{N}=6$) [60]	11.76	1.96×	924.30	4.02×	0.9830 (−1.0%)	32.25 (−1.1%)
`DuCa` ($\mathcal{N}=5$) [61]	7.32	3.15×	978.76	3.80×	0.9982 (+0.5%)	32.41 (−0.6%)
TeaCache ($l=0.8$) [22]	6.42	3.58×	892.35	4.17×	0.8710 (−12.3%)	31.89 (−2.2%)
DBCache ($\mathcal{F}=4, B=4$) [9]	5.92	3.90×	907.20	4.10×	0.6372 (−35.8%)	32.10 (−1.6%)
TaylorSeer ($\mathcal{N}=4, O=2$) [25]	8.31	2.80×	967.91	3.84×	0.9887 (−0.4%)	32.58 (−0.1%)
FoCa ($\mathcal{N}=4$) [56]	8.37	2.76×	1050.70	3.54×	0.9782 (−1.5%)	32.73 (+0.4%)
Clusca ($\mathcal{N}=4, O=2, K=16$) [58]	8.27	2.79×	1045.58	3.56×	0.9876 (−0.5%)	32.62 (+0.0%)
HyCa ($\mathcal{N}=5$) [55]	6.83	3.38×	893.54	4.16×	1.0096 (+1.7%)	32.87 (+0.8%)
LinCa ($\mathcal{N}=6$)	**5.27**	**4.38×**	**823.21**	**4.52×**	**1.0228** (+3.0%)	**32.97** (+1.1%)
FORA ($\mathcal{N}=5$) [41]	6.94	3.33×	893.54	4.16×	0.8276 (−16.7%)	31.95 (−2.0%)
`ToCa` ($\mathcal{N}=8$) [60]	10.17	2.27×	784.54	4.74×	0.9479 (−4.5%)	32.16 (−1.4%)
`DuCa` ($\mathcal{N}=7$) [61]	6.06	3.83×	760.14	4.89×	0.9785 (−1.5%)	32.26 (−1.1%)
TeaCache ($l=1$) [22]	7.24	3.19×	743.63	5.01×	0.8421 (−15.2%)	32.01 (−1.8%)
DBCache ($\mathcal{F}=4, B=2$) [9]	5.25	4.40×	793.07	4.69×	0.5106 (−48.6%)	32.01 (−1.8%)
TaylorSeer ($\mathcal{N}=5, O=2$) [25]	6.71	3.46×	893.54	4.16×	0.9793 (−1.4%)	32.63 (+0.1%)
FoCa ($\mathcal{N}=6$) [56]	6.76	3.42×	745.39	4.99×	0.9741 (−1.9%)	**33.10** (+1.5%)
Speca ($\mathcal{N}_{\max}=8, \mathcal{N}_{\min}=2$) [26]	6.61	3.48×	791.38	4.70×	1.0012 (+0.8%)	32.45 (−0.5%)
Clusca ($\mathcal{N}=5, O=1, K=16$) [58]	6.31	3.66×	897.03	4.14×	0.9748 (−1.8%)	32.50 (−0.3%)
HyCa ($\mathcal{N}=6$) [55]	6.09	3.79×	744.81	5.00×	1.0043 (+1.1%)	32.64 (+0.1%)
LinCa ($\mathcal{N}=8$)	**4.40**	**5.25×**	**674.64**	**5.51×**	**1.0162** (+2.3%)	32.72 (+0.3%)

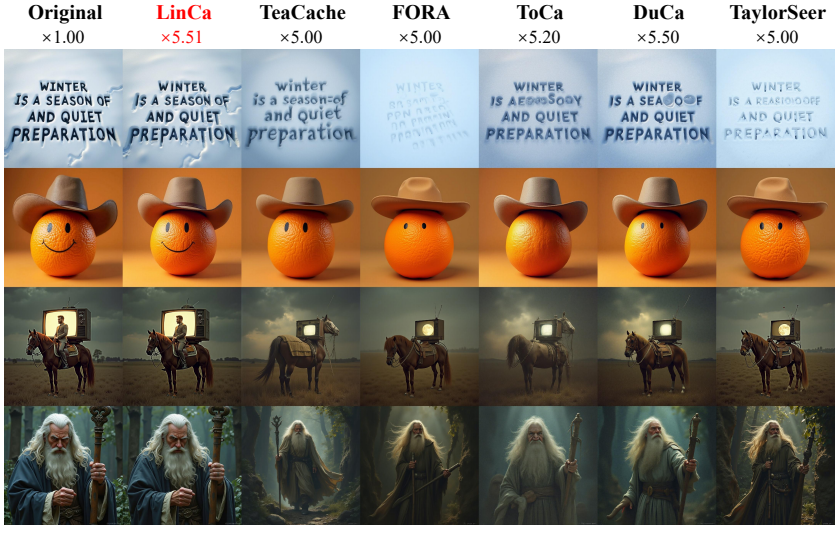

Fig. 4: On FLUX.1-dev, LinCa delivers higher speedup with better text consistency, fine-grained details, and spatial relationships.

Table 2: Quantitative comparison of text-to-image generation for Qwen-Image. Best results are highlighted in **bold**, and second-best are underlined.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	ImageReward↑	CLIP↑	PSNR↑	SSIM↑	LPIPS↓
Original: 50 steps	126.60	1.00×	12917.56	1.00×	1.2532 (+0.0%)	35.52	∞	1.00	0.00
50% steps	63.62	1.99×	6458.78	2.00×	1.2023 (−4.1%)	35.24	30.55	0.75	0.27
20% steps	25.89	4.89×	2583.51	5.00×	0.9223 (−26.4%)	34.94	28.60	0.60	0.52
TaylorSeer ($\mathcal{N}$=3) [25]	62.36	2.03×	**4646.60**	**2.78**×	1.0674 (−14.8%)	34.78	28.14	0.53	0.65
HyCa ($\mathcal{N}$=3) [55]	59.72	2.12×	**4646.60**	**2.78**×	1.2316 (−1.7%)	35.01	30.21	0.80	0.25
LinCa ($\mathcal{N}$=3)	**58.88**	**2.15**×	4705.76	2.75×	**1.2329 (−1.6%)**	**35.40**	**31.86**	**0.83**	**0.18**
FORA ($\mathcal{N}$=4) [41]	38.13	3.32×	3359.99	3.84×	0.9315 (−25.7%)	34.33	28.65	0.60	0.50
ToCa ($\mathcal{N}$=8, $\mathcal{R}$=75%) [60]	60.87	2.08×	2991.34	4.32×	1.0218 (−18.5%)	34.97	28.94	0.63	0.45
DuCa ($\mathcal{N}$=9, $\mathcal{R}$=80%) [61]	34.50	3.67×	2958.13	4.37×	0.7709 (−38.5%)	34.59	28.44	0.58	0.54
TaylorSeer ($\mathcal{N}$=6) [25]	30.58	4.14×	**2583.97**	**5.00**×	1.0125 (−19.2%)	34.77	28.59	0.62	0.45
HyCa ($\mathcal{N}$=6) [55]	36.48	3.47×	2584.46	5.00×	1.1967 (−4.5%)	34.89	29.68	0.71	0.31
LinCa ($\mathcal{N}$=6)	**30.51**	**4.15**×	2635.31	4.90×	**1.2163 (−2.9%)**	**35.34**	**29.84**	**0.73**	**0.29**
FORA ($\mathcal{N}$=6) [41]	28.51	4.44×	2326.74	5.55×	0.4812 (−61.6%)	33.31	28.48	0.55	0.59
ToCa ($\mathcal{N}$=12, $\mathcal{R}$=85%) [60]	50.64	2.50×	2406.20	5.37×	0.5511 (−56.0%)	34.05	28.68	0.58	0.54
DuCa ($\mathcal{N}$=12, $\mathcal{R}$=90%) [61]	28.39	4.46×	2171.56	5.95×	0.4104 (−67.3%)	33.34	28.38	0.57	0.61
TaylorSeer ($\mathcal{N}$=9) [25]	24.49	5.17×	2067.29	6.25×	0.7318 (−41.6%)	32.89	28.24	0.56	0.57
HyCa ($\mathcal{N}$=8) [55]	23.53	5.38×	2066.82	6.25×	1.0432 (−16.8%)	34.82	28.86	0.62	0.44
LinCa ($\mathcal{N}$=10)	**23.19**	**5.46**×	**1859.35**	**6.95**×	**1.0524 (−16.0%)**	**35.17**	**29.10**	**0.69**	**0.39**

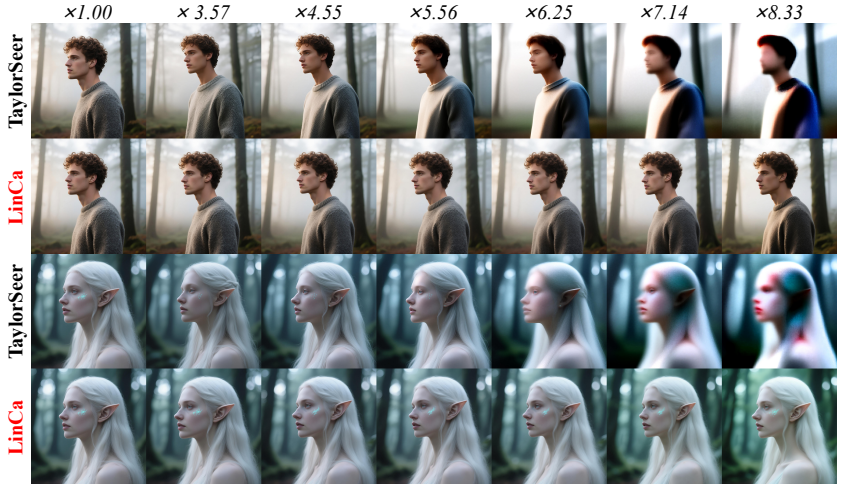

Fig. 5: On Qwen-Image, LinCa retains superior image quality and details as the acceleration ratio increases, while Taylorseer degrades with blurring and detail loss.

LinCa demonstrates remarkable robustness, achieving **1.0228** at **4.52**×, which outperforms both HyCa (1.0096 at 4.16×) and Clusca (0.9876 at 3.56×). At $N = 8$, it maintains a superior quality of **1.0162** at **5.51**×. Other methods such as DBCache and TeaCache suffer from substantial degradation. Visual comparison in Fig. 4 further confirms LinCa's advantage in preserving image details and text-alignment under high compression ratios.

In Table 2, LinCa consistently achieves the best overall trade-off on **Qwen-Image** across all acceleration levels. At $N = 3$, it is comparable to TaylorSeer in speed (**2.75**×) but yields higher quality (ImageReward **1.2329** vs. 1.0674, and highest PSNR **31.86**). At $N = 6$, LinCa remains strong (**1.2163**, **29.84**), outper-

forming HyCa (1.1967, 29.68) and surpassing other methods like ToCa (1.0218) and FORA (0.9315). At $N = 10$, it sustains high quality (**1.0524** at **6.95**×), while others drop sharply. These results highlight LinCa's robustness under high acceleration while preserving superior visual fidelity and text-alignment. Visual comparison in Fig. 5 further demonstrates that LinCa consistently preserves high image quality and intricate details even as the acceleration ratio increases.

4.3 Results on Text-to-Video Generation

Table 3: Quantitative comparison of text-to-video generation for HunyuanVideo. Best results are highlighted in **bold**, and second-best are underlined.

Method	Efficient Attention	Acceleration				VBench ↑ Score(%)
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	
Original: 50 steps	✔	185.00	1.00×	29773.0	1.00×	80.66 (+0.0%)
22% steps	✔	40.66	4.55×	6550.1	4.55×	78.74 (−2.4%)
FORA ($\mathcal{N} = 5$) [41]	✔	43.84	4.22×	5960.4	5.00×	78.83 (−2.3%)
`ToCa` ($\mathcal{N} = 5, R = 90\%$) [60]	✘	49.07	3.77×	7006.2	4.25×	78.86 (−2.2%)
`DuCa` ($\mathcal{N} = 5, R = 90\%$) [61]	✔	40.39	4.58×	6483.2	4.48×	78.72 (−2.4%)
TeaCache ($l = 0.4$) [22]	✔	<u>38.87</u>	<u>4.76</u>×	6550.1	4.55×	79.36 (−1.6%)
TaylorSeer ($\mathcal{N} = 5, O = 1$) [25]	✔	44.47	4.16×	5960.4	5.00×	79.93 (−0.9%)
Speca ($\mathcal{N}_{\max} = 8, \mathcal{N}_{\min} = 2$) [26]	✔	44.05	4.20×	<u>5692.7</u>	<u>5.23</u>×	79.98 (−0.8%)
Clusca ($\mathcal{N} = 5, K = 32$) [58]	✔	48.30	3.83×	5968.1	4.99×	<u>79.99</u> (−0.8%)
FoCa ($\mathcal{N} = 5$) [56]	✔	44.05	4.20×	5966.5	4.99×	79.96 (−0.9%)
LinCa ($\mathcal{N} = 6$)	✔	**38.21**	**4.84**×	**5413.27**	**5.50**×	**80.16** (−0.6%)

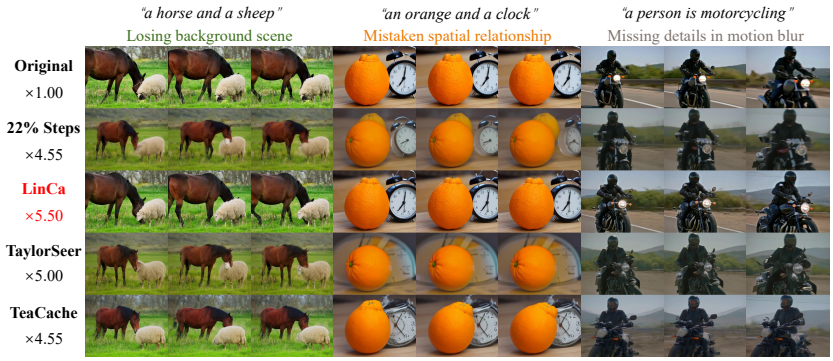

Fig. 6: On HunyuanVideo, LinCa maintains high-quality generation under higher acceleration ratio, while other methods suffer from lossing background scene, mistaken spatial relationships, and missing motional details.

As shown in Table 3, LinCa exhibits superior quality on **HunyuanVideo**. At $\mathcal{N} = 6$, it achieves a notable **5.50**× speedup while preserving a competitive VBench score of **80.16**, representing only a slight decrease from the 50-step baseline (80.66). In contrast, Clusca and Speca reach 79.99 and 79.98 with less speedup, while TaylorSeer reaches only 5.00× with 79.93 and ToCa/DuCa degrade further. This demonstrates an optimal balance between inference speed

and visual fidelity in high-quality video generation. Qualitative comparisons on Fig. 6 further confirm that LinCa preserves video quality, background scenery, spatial relationship and motional details better.

4.4 Results on Image Editing

Table 4: Quantitative comparison of image editing for Qwen-Image-Edit. Best results are highlighted in **bold**, and second-best results are underlined.

Method	Acceleration				GEdit-CN (Full)			GEdit-EN (Full)		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	SC ↑	PQ ↑	OS ↑	SC ↑	PQ ↑	OS ↑
Original: 50 steps	284.51	1.00×	28190.88	1.00×	7.68	7.51	7.41	7.82	7.54	7.54
50% **steps**	143.29	1.99×	14095.44	2.00×	7.70	7.53	7.44	7.77	7.52	7.47
20% **steps**	58.45	4.87×	5638.18	5.00×	7.65	7.42	7.35	7.73	7.46	7.44
FORA ($\mathcal{N}$=5) [41]	63.15	4.51×	5643.13	5.00×	7.60	7.31	7.25	7.62	7.34	7.28
DuCa ($\mathcal{N}$=7, $\mathcal{R}$=95%) [61]	69.54	4.09×	5699.89	4.95×	**7.73**	7.44	7.44	7.80	7.40	7.45
TaylorSeer ($\mathcal{N}$=6) [25]	65.66	4.33×	5643.13	5.00×	7.25	7.09	6.92	7.26	7.14	6.89
LinCa ($\mathcal{N}$=7)	**60.02**	**4.74×**	**5110.58**	**5.52×**	7.63	**7.52**	**7.45**	**7.81**	**7.55**	**7.56**
FORA ($\mathcal{N}$=7) [41]	52.20	5.45×	4515.74	6.24×	7.42	7.13	7.06	7.43	7.19	7.06
DuCa ($\mathcal{N}$=10, $\mathcal{R}$=95%) [61]	59.81	4.76×	5158.45	5.46×	7.50	5.75	6.39	7.52	5.77	6.41
TaylorSeer ($\mathcal{N}$=9) [25]	53.92	5.28×	4515.74	6.24×	6.61	6.65	6.31	6.67	6.63	6.31
LinCa ($\mathcal{N}$=10)	**49.05**	**5.80×**	**3982.82**	**7.08×**	**7.55**	**7.40**	**7.27**	**7.68**	**7.46**	**7.40**

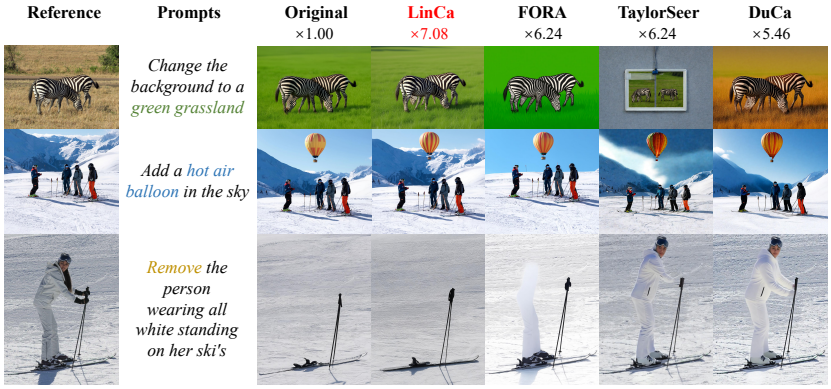

Fig. 7: On Qwen-Image-Edit, LinCa delivers better prompt comprehension and generates high-fidelity images while maintaining consistency in non-edited regions.

As shown in Table 4, LinCa exhibits exceptional performance on **Qwen-Image-Edit**. At $N = 7$, it obtains overall scores of **7.45** (CN) and **7.56** (EN), outperforming TaylorSeer (6.92/6.89), FORA (7.25/7.28), and DuCa (7.44/7.45), even exceeding the original model's performance. At $N = 10$, LinCa continues to lead with **7.27**/**7.40**, whereas other baselines such as DuCa and TaylorSeer drop to 6.39/6.41 and 6.31/6.31. These results demonstrate that LinCa remains robust under high acceleration, effectively preserving high-fidelity synthesis. Qualitative comparisons on Fig. 7 further confirm its effectiveness in handling different editing tasks with high quality and consistency in non-edited regions.

4.5 Results on Distilled or Quantized Models

Table 5: Quantitative comparison of text-to-image generation for FLUX.1-lite-8B, FLUX.1-schnell and FLUX.1-dev-int8.

Method	Acceleration				ImageReward↑	CLIP Score↑
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
FLUX.1-lite-8B: 28 steps	8.21	1.00×	1291.49	1.00×	0.8936 (+0.0%)	32.12 (+0.0%)
LinCa ($\mathcal{N}$=3): 28 steps	**3.34**	**2.46×**	**556.21**	**2.32×**	**0.9070** (+1.5%)	**32.36** (+0.7%)
FLUX.1-schnell: 4 steps	2.48	1.00×	283.56	1.00×	0.9692 (+0.0%)	32.54 (+0.0%)
LinCa ($\mathcal{N}$=3): 4 steps	**1.38**	**1.80×**	**142.16**	**1.99×**	**0.9843** (+1.6%)	**32.67** (+0.4%)
FLUX.1-dev-int8: 50 steps	12.55	1.00×	1888.07	1.00×	0.9744 (+0.0%)	32.55 (+0.0%)
LinCa ($\mathcal{N}$=3): 50 steps	**4.61**	**2.72×**	**718.17**	**2.63×**	**1.0036** (+3.0%)	**32.81** (+0.8%)

To validate the generality of LinCa, we evaluate its performance when integrated with other mainstream acceleration techniques, specifically model distillation (**FLUX.1-lite-8B**), step distillation (**FLUX.1-schnell**), and quantization (**FLUX.1-dev-int8**). As shown in Table 5, LinCa consistently improves performance across all settings without degrading generation quality. For the model-distilled variant, LinCa achieves a **2.32**× speed-up, increasing the ImageReward to **0.9070** and the CLIP Score to **32.36**. When applied to the step-distilled model, our framework delivers a **1.99**× acceleration, reaching an **0.9843** ImageReward and **32.67** CLIP Score. Furthermore, integrating LinCa with INT8 quantization provides a **2.63**× speed improvement while maintaining high fidelity (**1.0036** ImageReward, **32.81** CLIP Score). These results demonstrate the broad compatibility of LinCa with various mainstream acceleration methods, serving as an effective complementary module that further enhances acceleration performance while maintaining high generation quality.

4.6 Ablation Study

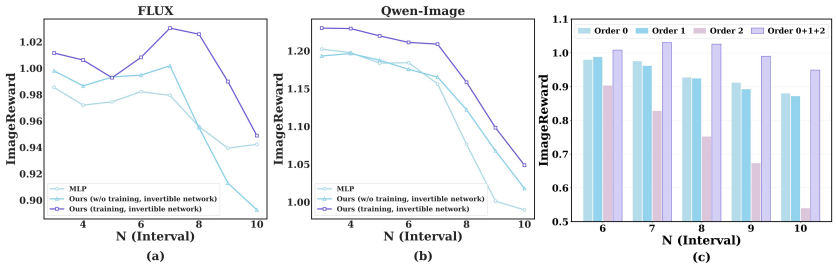

Fig. 8: Ablation results of LinCa. (a)-(b): Comparison of decomposition network architectures on FLUX.1-dev and Qwen-Image. The learnable invertible network consistently outperforms both MLP and untrained alternatives. **(c)** Comparison of prediction order strategies. Differentiated multi-order prediction maintains superior quality over single-order prediction.

Decomposition Network Ablation. We study the impact of different decomposition networks on generation quality on FLUX.1-dev and Qwen-Image. As shown in Figure 8(a)-(b), LinCa's learnable invertible network architecture achieves the best results across all intervals. Compared to ordinary non-invertible networks such as learnable MLPs, the invertible network yields higher overall quality, indicating that its lossless reconstruction property effectively avoids information loss during the mapping process. Compared to non-learnable strategies that exhibit significant quality degradation under high acceleration ratios, the learnable approach demonstrates more stable performance, adapting to the specific feature dynamics.

Prediction Strategy Ablation. We further compare using 0^{th} order (pure reuse), 1^{st} order, and 2^{nd} order prediction individually against LinCa's combined $0^{th} + 1^{st} + 2^{nd}$ order prediction, to investigate the impact of different prediction strategies on generation quality. As shown in Figure 8(c), LinCa's differentiated order prediction strategy outperforms single-order configurations across all intervals. Notably, as N increases, uniform single-order prediction, especially 2^{nd} order prediction alone, suffers from significant quality degradation, while LinCa accurately separates features into sub-components with different continuity through learnable mapping and applies differentiated prediction, consistently maintaining the highest generation quality.

Furthermore, since the feature dynamics differ across timestep segments, LinCa trains isomorphic but separately parameterized predictors for each segment. Experiments show that partitioning into $S = 3$ segments already yields good results, and further increasing S brings diminishing improvements. Detailed results, additional ablation studies, and more hyperparameter analyses are provided in the supplementary materials.

5 Conclusion

This paper proposes ***LinCa***, a feature caching acceleration framework based on learnable invertible networks. LinCa reveals that the hidden features of diffusion models exhibit significant dynamics differences across denoising stages, models, and feature dimensions. Based on it, LinCa decomposes cached features into sub-components with distinct continuity via a lightweight invertible network and applies differentiated polynomial prediction. Through data-driven per-segment training, LinCa adaptively accommodates the feature dynamics of different models and denoising stages, achieving a superior trade-off between acceleration and generation quality. Extensive experiments on FLUX, Qwen-Image, and HunyuanVideo demonstrate that LinCa significantly outperforms existing methods at the same acceleration ratio, achieving $5-7\times$ speedup with minimal quality degradation, while remaining compatible with distilled or quantized models. We believe LinCa opens up new possibilities for scalable, high-performance generative models and provides a new direction for efficient diffusion inference.

Acknowledgements

This paper was partially sponsored by Terminal Intelligent Computing Division - Alibaba Cloud.

References

1. Black Forest Labs: Flux.1-schnell. https://huggingface.co/black-forest-labs/FLUX.1-schnell (2024), hugging Face model card
2. Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al.: Stable video diffusion: Scaling latent video diffusion models to large datasets. arXiv preprint arXiv:2311.15127 (2023)
3. Bolya, D., Hoffman, J.: Token merging for fast stable diffusion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4599–4603 (2023)
4. Cai, P., Liu, J., Xu, H., Wang, X., Zou, C., Zhang, L.: Lesa: Learnable stage-aware predictors for diffusion model acceleration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 43300–43309 (2026)
5. Chen, J., Ge, C., Xie, E., Wu, Y., Yao, L., Ren, X., Wang, Z., Luo, P., Lu, H., Li, Z.: Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation (2024)
6. Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wu, Y., Wang, Z., Kwok, J., Luo, P., Lu, H., Li, Z.: Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In: International Conference on Learning Representations (2024)
7. Chen, P., Shen, M., Ye, P., Cao, J., Tu, C., Bouganis, C.S., Zhao, Y., Chen, T.: δ-dit: A training-free acceleration method tailored for diffusion transformers. arXiv preprint arXiv:2406.01125 (2024)
8. Cheng, X., Chen, Z., Jia, Z.: Cat pruning: Cluster-aware token pruning for text-to-image diffusion models (2025), https://arxiv.org/abs/2502.00433
9. DefTruth, v.: cache-dit: A pytorch-native and flexible inference engine with hybrid cache acceleration and parallelism for dits. (2025), https://github.com/vipshop/cache-dit.git, open-source software available at https://github.com/vipshop/cache-dit.git
10. Diffusers: Flux.1-dev-torchao-int8. https://huggingface.co/diffusers/FLUX.1-dev-torchao-int8 (2024), hugging Face model card; quantized checkpoint derived from black-forest-labs/FLUX.1-dev
11. Fang, G., Ma, X., Wang, X.: Structural pruning for diffusion models. arXiv preprint arXiv:2305.10924 (2023)
12. Freepik: Flux.1 lite: Distilling flux1.dev for efficient text-to-image generation. https://huggingface.co/Freepik/flux.1-lite-8B (2024), hugging Face model card; page includes citation to an article (Verdú and Martín, 2024)
13. Gomez, A.N., Ren, M., Urtasun, R., Grosse, R.B.: The reversible residual network: Backpropagation without storing activations. Advances in neural information processing systems **30** (2017)
14. Hessel, J., Holtzman, A., Forbes, M., Bras, R.L., Choi, Y.: Clipscore: A reference-free evaluation metric for image captioning. arXiv preprint arXiv:2104.08718 (2021)

15. Ho, J., Jain, A., Abbeel, P.: Denoising Diffusion Probabilistic Models (Dec 2020). https://doi.org/10.48550/arXiv.2006.11239, http://arxiv.org/abs/2006.11239, arXiv:2006.11239 [cs]
16. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
17. Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., Liu, Z.: VBench: Comprehensive Benchmark Suite for Video Generative Models (Nov 2023). https://doi.org/10.48550/arXiv.2311.17982, http://arxiv.org/abs/2311.17982, arXiv:2311.17982 [cs]
18. Kim, M., Gao, S., Hsu, Y.C., Shen, Y., Jin, H.: Token fusion: Bridging the gap between token pruning and token merging. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1383–1392 (2024)
19. Kim, S., Lee, H., Cho, W., Park, M., Ro, W.W.: Ditto: Accelerating diffusion model via temporal value similarity. In: Proceedings of the 2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE (2025)
20. Labs, B.F.: Flux. https://github.com/black-forest-labs/flux (2024)
21. Li, Y., Wang, H., Jin, Q., Hu, J., Chemerys, P., Fu, Y., Wang, Y., Tulyakov, S., Ren, J.: Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. Advances in Neural Information Processing Systems **36** (2024)
22. Liu, F., Zhang, S., Wang, X., Wei, Y., Qiu, H., Zhao, Y., Zhang, Y., Ye, Q., Wan, F.: Timestep embedding tells: It's time to cache for video diffusion model (2024)
23. Liu, J., Cai, P., Zhou, Q., Lin, Y., Kong, D., Huang, B., Pan, Y., Xu, H., Zou, C., Tang, J., et al.: Freqca: Accelerating diffusion models via frequency-aware caching. arXiv preprint arXiv:2510.08669 (2025)
24. Liu, J., Wang, X., Lin, Y., Wang, Z., Wang, P., Cai, P., Zhou, Q., Yan, Z., Yan, Z., Shi, Z., et al.: A survey on cache methods in diffusion models: Toward efficient multi-modal generation. arXiv preprint arXiv:2510.19755 (2025)
25. Liu, J., Zou, C., Lyu, Y., Chen, J., Zhang, L.: From reusing to forecasting: Accelerating diffusion models with taylorseers (2025), https://arxiv.org/abs/2503.06923
26. Liu, J., Zou, C., Lyu, Y., Li, K., Wang, S., Zhang, L.: Speca: Accelerating diffusion transformers with speculative feature caching. In: Proceedings of the 33rd ACM International Conference on Multimedia (MM '25). p. to appear. ACM, Dublin, Ireland (October 2025)
27. Liu, S., Han, Y., Xing, P., Yin, F., Wang, R., Cheng, W., Liao, J., Wang, Y., Fu, H., Han, C., Li, G., Peng, Y., Sun, Q., Wu, J., Cai, Y., Ge, Z., Ming, R., Xia, L., Zeng, X., Zhu, Y., Jiao, B., Zhang, X., Yu, G., Jiang, D.: Step1x-edit: A practical framework for general image editing (2025), https://arxiv.org/abs/2504.17761
28. Liu, X., Gong, C., et al.: Flow straight and fast: Learning to generate and transfer data with rectified flow. In: The Eleventh International Conference on Learning Representations (2023)
29. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: 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)
30. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. arXiv preprint arXiv:2211.01095 (2022)
31. Ma, Y., Feng, K., Zhang, X., Liu, H., Zhang, D.J., Xing, J., Zhang, Y., Yang, A., Wang, Z., Chen, Q.: Follow-your-creation: Empowering 4d creation through video inpainting. arXiv preprint arXiv:2506.04590 (2025)

32. Ma, Y., He, Y., Cun, X., Wang, X., Chen, S., Li, X., Chen, Q.: Follow your pose: Pose-guided text-to-video generation using pose-free videos. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 4117–4125 (2024)
33. Ma, Y., Liu, Y., Zhu, Q., Yang, A., Feng, K., Zhang, X., Li, Z., Han, S., Qi, C., Chen, Q.: Follow-your-motion: Video motion transfer via efficient spatial-temporal decoupled finetuning. arXiv preprint arXiv:2506.05207 (2025)
34. Ma, Y., Wang, X., Ma, Q., Wang, Q., Zheng, M., Yang, X., Li, H., Zhao, C., Ying, J., Yang, H., et al.: Group editing: Edit multiple images in one go. arXiv preprint arXiv:2603.22883 (2026)
35. Ma, Y., Wang, Z., Ren, T., Zheng, M., Liu, H., Guo, J., Fong, M., Xue, Y., Zhao, Z., Schindler, K., et al.: Fastvmt: Eliminating redundancy in video motion transfer. arXiv preprint arXiv:2602.05551 (2026)
36. Meng, C., Gao, R., Kingma, D.P., Ermon, S., Ho, J., Salimans, T.: On distillation of guided diffusion models. In: NeurIPS 2022 Workshop on Score-Based Methods (2022), https://openreview.net/forum?id=6QHpSQt6VR-
37. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4195–4205 (2023)
38. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models (Apr 2022). https://doi.org/10.48550/arXiv.2112.10752, http://arxiv.org/abs/2112.10752, arXiv:2112.10752 [cs]
39. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. https://doi.org/10.48550/arXiv.2205.11487, http://arxiv.org/abs/2205.11487
40. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
41. Selvaraju, P., Ding, T., Chen, T., Zharkov, I., Liang, L.: Fora: Fast-forward caching in diffusion transformer acceleration. arXiv preprint arXiv:2407.01425 (2024)
42. Shang, Y., Yuan, Z., Xie, B., Wu, B., Yan, Y.: Post-training quantization on diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1972–1981 (2023)
43. Silveria, A., Govande, S.V., Fu, D.Y.: Chipmunk: Training-free acceleration of diffusion transformers with dynamic column-sparse deltas. In: ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models (2025)
44. Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics (2015), https://arxiv.org/abs/1503.03585
45. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021)
46. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. In: International Conference on Machine Learning. pp. 32211–32252. PMLR (2023)
47. Sun, X., Chen, Y., Huang, et al.: Hunyuan-large: An open-source MoE model with 52 billion activated parameters by tencent. https://doi.org/10.48550/arXiv.2411.02265, http://arxiv.org/abs/2411.02265
48. Wu, C., Li, J., Zhou, J., Lin, J., Gao, K., Yan, K., ming Yin, S., Bai, S., Xu, X., Chen, Y., Chen, Y., Tang, Z., Zhang, Z., Wang, Z., Yang, A., Yu, B., Cheng, C., Liu, D., Li, D., Zhang, H., Meng, H., Wei, H., Ni, J., Chen, K., Cao, K., Peng,

L., Qu, L., Wu, M., Wang, P., Yu, S., Wen, T., Feng, W., Xu, X., Wang, Y., Zhang, Y., Zhu, Y., Wu, Y., Cai, Y., Liu, Z.: Qwen-image technical report (2025), https://arxiv.org/abs/2508.02324
49. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation (Dec 2023). https://doi.org/10.48550/arXiv.2304.05977, http://arxiv.org/abs/2304.05977, arXiv:2304.05977 [cs]
50. Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., Yin, D., Gu, X., Yuxuan.Zhang, Wang, W., Cheng, Y., Xu, B., Dong, Y., Tang, J.: Cogvideox: Text-to-video diffusion models with an expert transformer. In: The Thirteenth International Conference on Learning Representations (2025), https://openreview.net/forum?id=LQzN6TRFg9
51. Yuan, Z., Zhang, H., Lu, P., Ning, X., Zhang, L., Zhao, T., Yan, S., Dai, G., Wang, Y.: Ditfastattn: Attention compression for diffusion transformer models. arXiv preprint arXiv:2406.08552 (2024)
52. Zhang, E., Tang, J., Ning, X., Zhang, L.: Training-free and hardware-friendly acceleration for diffusion models via similarity-based token pruning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)
53. Zhang, E., Xiao, B., Tang, J., Ma, Q., Zou, C., Ning, X., Hu, X., Zhang, L.: Token pruning for caching better: 9 times acceleration on stable diffusion for free (2024), https://arxiv.org/abs/2501.00375
54. Zhao, X., Jin, X., Wang, K., You, Y.: Real-time video generation with pyramid attention broadcast. arXiv preprint arXiv:2408.12588 (2024)
55. Zheng, S., Chen, G., Zhou, Q., Lin, Y., He, L., Zou, C., Cai, P., Liu, J., Zhang, L.: Let features decide their own solvers: Hybrid feature caching for diffusion transformers. arXiv preprint arXiv:2510.04188 (2025)
56. Zheng, S., Feng, L., Wang, X., Zhou, Q., Cai, P., Zou, C., Liu, J., Lin, Y., Chen, J., Ma, Y., Zhang, L.: Forecast then calibrate: Feature caching as ode for efficient diffusion transformers (2025). https://doi.org/10.48550/arXiv.2508.16211
57. Zheng, Z., Peng, X., Yang, T., Shen, C., Li, S., Liu, H., Zhou, Y., Li, T., You, Y.: Open-sora: Democratizing efficient video production for all (March 2024), https://github.com/hpcaitech/Open-Sora
58. Zheng, Z., Wang, X., Zou, C., Wang, S., Zhang, L.: 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). p. to appear. ACM, Dublin, Ireland (October 2025)
59. Zhu, H., Tang, D., Liu, J., Lu, M., Zheng, J., Peng, J., Li, D., Wang, Y., Jiang, F., Tian, L., Tiwari, S., Sirasao, A., Yong, J.H., Wang, B., Barsoum, E.: Dip-go: A diffusion pruner via few-step gradient optimization (2024)
60. Zou, C., Liu, X., Liu, T., Huang, S., Zhang, L.: Accelerating diffusion transformers with token-wise feature caching. In: Proceedings of the 13th International Conference on Learning Representations (ICLR 2025). ICLR (2025), https://openreview.net/forum?id=yYZbZGo4ei, accepted to ICLR 2025
61. Zou, C., Zhang, E., Guo, R., Xu, H., He, C., Hu, X., Zhang, L.: Accelerating diffusion transformers with dual feature caching (2024), https://arxiv.org/abs/2412.18911