LET FEATURES DECIDE THEIR OWN SOLVERS: HYBRID FEATURE CACHING FOR DIFFUSION TRANSFORMERS

Shikang Zheng[1,2] **Guantao Chen**[1] **Qinming Zhou**[1,3] **Yuqi Lin**[1] **Lixuan He**[1,3]
Chang Zou[1] **Peiliang Cai**[1] **Jiacheng Liu**[1] **Linfeng Zhang**[1†]
[1]Shanghai Jiao Tong University [2]South China University of Technology [3]Tsinghua University

ABSTRACT

Diffusion Transformers offer state-of-the-art fidelity in image and video synthesis, but their iterative sampling process remains a major bottleneck due to the high cost of transformer forward passes at each timestep. To mitigate this, feature caching has emerged as a training-free acceleration technique that reuses or forecasts hidden representations. However, existing methods often apply a uniform caching strategy across all feature dimensions, ignoring their heterogeneous dynamic behaviors. Therefore, we adopt a new perspective by modeling hidden feature evolution as a mixture of ODEs across dimensions, and introduce **HyCa**, a Hybrid ODE solver inspired caching framework that applies dimension-wise caching strategies. HyCa achieves near-lossless acceleration across diverse domains and models, including 5.55× speedup on FLUX, 5.56× speedup on HunyuanVideo, 6.24× speedup on Qwen-Image and Qwen-Image-Edit without retraining.

1 INTRODUCTION

Figure 1: Images generated on Qwen-Image with HyCa at **6.24**× acceleration.

Diffusion Transformers (DiTs) have recently achieved impressive success across image and video generation tasks, demonstrating strong modeling capacity and generation quality. However, the iterative nature of diffusion sampling presents a significant bottleneck, as each output demands multiple transformer passes. This high computational cost hinders deployment in scenarios with strict latency or resource constraints, driving the ongoing research on efficient inference methods.

To address this challenge, two primary acceleration directions have emerged: reducing the total number of sampling steps via algorithmic advancements (Lu et al., 2022a), and lowering the cost of each step through architectural optimization (Yuan et al., 2024a; Zhao et al., 2024). Among these,

†Corresponding author.

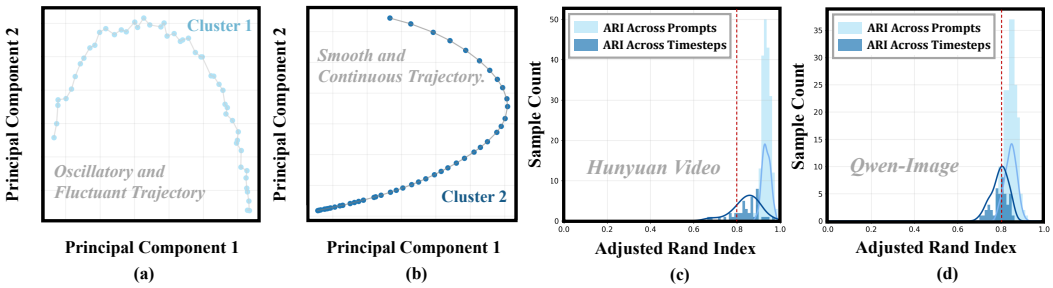

Figure 2: **Feature trajectory clusters and stability of assignments.** (a–b) Cluster 1 shows oscillatory trajectories while Cluster 2 shows smooth ones. (c–d) ARI distributions on Hunyuan Video and Qwen-Image exceed 0.8 in most cases, confirming stable and consistent cluster assignments across prompts and timesteps. An ARI above 0.8 indicates strong agreement and high clustering reliability.

training-free feature caching has emerged as a promising solution. It exploits the temporal coherence of hidden representations by reusing features, thereby reducing redundant computation. Early works such as DeepCache (Ma et al., 2024) demonstrated the feasibility of this idea in U-Net backbones, recent methods such as FORA (Selvaraju et al., 2024), ToCa (Zou et al., 2024a), TaylorSeer (Liu et al., 2025a) extended caching to transformer-based architectures and showed that feature caching can be effectively viewed as solving the temporal evolution of hidden features. Despite progress, current approaches are still limited in critical ways.

Existing methods implicitly assume that all hidden dimensions evolve under a single, unified system. However, this assumption is untenable in DiTs, where the feature space is high-dimensional and exhibits complicated behaviors. Such complexity is unlikely to be captured by a single process. To further investigate, we analyze how each feature dimension changes over timesteps and group them into clusters based on their dynamics. As shown in Fig. 2(a), some dimensions fluctuate sharply with oscillatory patterns, indicating stiffness or multimodal behavior, while others evolve smoothly and predictably, reflecting stable dynamics on Fig. 2(b). These observations suggest that the feature space of DiTs is better described as a complex system, where different groups of dimensions follow distinct temporal patterns, highlighting the need for tailored solvers rather than a one-size-fits-all approach.

Therefore, we introduce **HyCa**, a hybrid caching framework that models hidden feature evolution as a mixture of ODEs and applies suitable solvers for every dimension. Hyca begins with unsupervised clustering, grouping dimensions with similar dynamic behaviors, and modeling them into a shared ODE. Then, HyCa assigns the most suitable solver to each cluster. Normally, identifying the best solver would require running inference on a large set of images and comparing quantitative metrics. However, surprisingly, we found that cluster assignments are highly stable across resolutions, timesteps, and even prompts. As shown in Fig. 2(c)(d), this invariance allows us to evaluate solver performance on a **single prompt** at a **single timestep** to reliably identify the best solver, achieving results comparable to large-scale evaluation. Thus, with *"One-Time Choosing"* performed offline for each model, *"All-Time Solving"* becomes possible without any additional cost during inference.

HyCa provides robust and adaptive feature prediction across diverse tasks and architectures. Without retraining, it achieves near-lossless acceleration of **5.56**× on FLUX and Hunyuan Video, **6.24**× on Qwen-Image and Qwen-Image-Edit. Moreover, it is also fully compatible with distillation, reaching up to **24.4**× speedup on FLUX and **12.2**× on Qwen-Image while maintaining strong image quality. In summary, our main contributions are:

- **Heterogeneous Feature Dynamics.** We show that feature dimensions in DiTs do not follow a single unified system but exhibit heterogeneous dynamic behaviors that are better described as a mixture of ODEs. Through dynamics clustering analysis across multiple settings, we further reveal that these cluster's distributions are consistent and input-invariant.
- **HyCa Framework.** Inspired by hybrid ODE solvers in numerical analysis, we propose HyCa, a training-free framework that groups feature dimensions by their dynamics and automatically assigns the most suitable solver to each group, with minimal overhead.
- **Outstanding Performance.** We evaluate HyCa across diverse architectures and tasks, including Drawbench on FLUX and Qwen-Image, Vbench on HunyuanVideo, GEdit-Bench on Qwen-Image-Edit, and even distilled models. In all settings, HyCa delivers state-of-the-art performance.

2 RELATED WORK

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have achieved strong image/video generation quality. Early U-Net backbones (Ronneberger et al., 2015) faced scaling limits that Diffusion Transformers (DiT) (Peebles & Xie, 2023b) alleviated, enabling rapid progress across modalities and resolutions (Chen et al., 2024b;a; Zheng et al., 2024; Yang et al., 2025). Nevertheless, the iterative nature of sampling remains a key inference bottleneck. Two complementary research lines thus emerge: (i) reducing the number of steps and (ii) reducing the cost per step. Beyond speed, a central challenge is maintaining stability and fidelity under aggressive acceleration, especially when feature dynamics are heterogeneous across dimensions and timesteps.

2.1 SAMPLING TIMESTEP REDUCTION

DDIM (Song et al., 2021) introduced deterministic few-step sampling that preserves perceptual quality. Higher-order ODE solvers (DPM-Solver and variants) (Lu et al., 2022a;b; Zheng et al., 2023) improve accuracy–cost trade-offs via multi-step/multi-stage discretizations with carefully controlled local truncation error. Rectified Flow (Liu et al., 2023a) shortens transport paths, while distillation (Salimans & Ho, 2022) compresses long trajectories into compact generators. Consistency models (Song et al., 2023) enable few-step synthesis by learning a direct noise-to-clean mapping.

2.2 DENOISING NETWORK ACCELERATION

Model Compression Acceleration. Pruning (Fang et al., 2023; Zhu et al., 2024), quantization (Li et al., 2023b; Shang et al., 2023; Kim et al., 2025), distillation (Li et al., 2024), and token reduction (Bolya & Hoffman, 2023; Kim et al., 2024; Zhang et al., 2024; 2025; Cheng et al., 2025) reduce compute with limited runtime overhead. While effective, they typically require additional training and may degrade robustness under domain shifts if the compression is too aggressive.

Feature Caching Acceleration. Feature caching reuses activations to avoid redundant computation. Early U-Net methods (Li et al., 2023a; Ma et al., 2024) inspired DiT-specific designs: FasterCache (Lv et al., 2025), FORA (Selvaraju et al., 2024), Δ-DiT (Chen et al., 2024c), TeaCache (Liu et al., 2024), and DiTFastAttn (Yuan et al., 2024b). Dynamic updates (ToCa/DuCa) (Zou et al., 2024a;b), unified cache–prune pipelines (Sun et al., 2025), and region-adaptive sampling (Liu et al., 2025c) further improve efficiency. Among these advances, TaylorSeer (Liu et al., 2025a) exemplifies the *cache-then-forecast* paradigm by polynomial extrapolation from cached neighbors.

3 METHOD

3.1 PRELIMINARY

Diffusion Models. Diffusion models (Ho et al., 2020; Song et al., 2021) generate structured data by progressively refining random noise through a series of denoising steps. At each timestep t, the model predicts a conditional Gaussian distribution over x_{t-1} given x_t, where both the mean and variance are parameterized. This generative process can be formulated as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\tau_\theta(x_t, t)\right), \beta_t \mathbf{I}\right), \tag{1}$$

where $\mathcal{N}$ denotes a normal distribution, α_t and β_t are noise schedule parameters, and $\tau_\theta(x_t, t)$ denotes the model's estimate of the noise component. Sampling begins from a pure noise vector and proceeds by repeatedly drawing samples from these intermediate distributions until a clean image is produced.

Diffusion Transformer Architecture. The Diffusion Transformer (DiT) (Peebles & Xie, 2023a) adopts a hierarchical design, expressed as a composition of modules $\mathcal{G} = g_1 \circ g_2 \circ \cdots \circ g_L$. Each module g_l consists of a self-attention layer ($\mathcal{F}^l_{\text{SA}}$), a cross-attention layer ($\mathcal{F}^l_{\text{CA}}$), and a feedforward MLP ($\mathcal{F}^l_{\text{MLP}}$). These components are dynamically modulated across timesteps to accommodate the evolving noise levels during generation. The input $\mathbf{x}_t = \{x_i\}_{i=1}^{H\times W}$ is represented as a sequence of patch tokens. Each module includes a residual update of the form $\mathcal{F}(\mathbf{x}) = \mathbf{x} + \text{AdaLN} \circ f(\mathbf{x})$, where AdaLN (adaptive layer normalization) conditions the normalization parameters on the noise timestep, allowing for more effective denoising across varying noise scales.

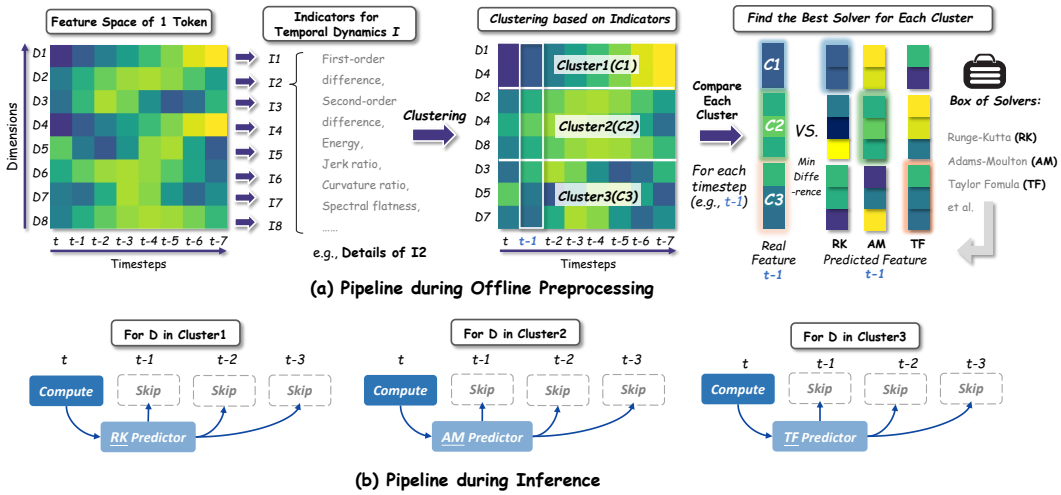

Figure 3: **HyCa Framework.** (a) Offline Preprocessing: feature dimensions are first analyzed and clustered with temporal indicators (e.g., differences, curvature). For each cluster, candidate solvers generate predicted features, then compared against real computed features; the solver with minimum error is then assigned to that cluster. (b) Inference: once assigned, each cluster consistently reuses its solver, enabling efficient prediction by skipping redundant computations while maintaining accuracy.

Feature Caching. Feature caching aims to reduce the cost of diffusion sampling by avoiding repeated computation of hidden features across timesteps. At each timestep t, the model produces hidden features $\mathcal{F}_t = \{\mathcal{F}_t^l\}_{l=1}^L$, and a caching function $\mathcal{C}(\mathcal{F}_A, k)$ estimates features $\tilde{\mathcal{F}}_k$ at a future timestep k using cached features from earlier timesteps. A common strategy is to reuse features from the last computed step:

$$\tilde{\mathcal{F}}_k = \mathcal{C}(\mathcal{F}_t, k) := \mathcal{F}_t, \quad \forall k \in (t, t+n-1], \tag{2}$$

which provides up to $(n-1)\times$ speedup. Recent methods improve reuse by forecasting future features, yet their reliance on a uniform prediction strategy across all dimensions often proves unstable in DiT's complex hidden feature space. In this work, we propose a hybrid approach that assigns suitable solvers for every dimension according to their dynamic behaviors.

3.2 Feature Caching as Hybrid ODE Solving

During reverse-time denoising in diffusion models, the hidden features evolve across timesteps. Let x_t be the latent variable at time t, and let $\mathcal{F}(x_t)$ denote the hidden feature extracted from it. Since the generative model is differentiable and x_t follows a continuous reverse-time trajectory, the composite feature map $t \mapsto \mathcal{F}(x_t)$ is also differentiable. By the chain rule and the probability flow ODE governing x_t, the feature dynamics satisfy:

$$\frac{d}{dt}\mathcal{F}(x_t) = g_\theta\big(\mathcal{F}(x_t),\, t\big), \tag{3}$$

where g_θ captures the implicit time-dependent vector field induced by the underlying network weights and structure. Although g_θ is not directly accessible in closed-form, we can sample the trajectory $\{\mathcal{F}(x_{t_k})\}$ on a discrete timestep grid, enabling numerical integration using only cached feature values. This perspective naturally casts feature caching as a numerical ODE solving problem. Instead of performing full forward computation at every timestep, we aim to solve the next feature value using prior ones:

$$\hat{\mathcal{F}}_{t+1} \approx \text{Solver}(\mathcal{F}_t, \mathcal{F}_{t-1}, \dots)$$

To accommodate diverse local feature dynamics, from smooth near-linear segments to rapidly varying regions, we adopt a hybrid solver strategy. Concretely, we define a predictor pool $\mathcal{S}$ that includes both explicit and implicit numerical solvers with different stability and accuracy properties, including Runge-Kutta(RK), Adams-Bashforth(AB), Taylor Fomula(TF), Backward Differentiation Formula(BDF) and Adams-Moulton(AM). This diverse solver set enables HyCa to assign methods tailored to local feature dynamics. *Please refer to A.2.2 for detailed mathematical formulations.*

3.3 HyCa Framework

Building on this foundation, **HyCa** is designed as a feature caching framework that models hidden dynamics as a mixture of ODEs and automatically assigns the most suitable solver to each cluster through a one-time optimization procedure. HyCa begins by analyzing the temporal dynamic behavior of each feature dimension. During a probe pass on a single prompt at the first few timesteps, we extract a descriptor vector $\phi_d \in \mathbb{R}^k$ for each feature dimension $d \in \{1, \ldots, D\}$, capturing dynamic indicators such as Jerk ratio and curvature ratio. Then, we apply k-means clustering to obtain a partition $\{c(d)\}$, where each dimension d is assigned to a cluster $c \in \{1, \ldots, C\}$. These cluster assignments remain stable across prompts, timesteps and resolutions, thus reused throughout inference.

The resulting clusters represent groups of dimensions that share similar temporal behaviors, enabling solver assignments to be conducted independently for each cluster. Given a solver pool $\mathcal{S}$, HyCa selects the optimal solver $s_c^\star \in \mathcal{S}$ for each cluster c by minimizing the average next-step prediction error across all dimensions in that cluster:

$$\min_{\{s_c \in \mathcal{S}\}_{c=1}^{C}} \sum_{c=1}^{C} \left[\frac{1}{|c|} \sum_{d \in c} \left\| \hat{\mathcal{F}}_{t+1}^{(s_c,d)} - \mathcal{F}_{t+1}^{(d)} \right\|_2^2 \right], \tag{4}$$

where $\hat{\mathcal{F}}_{t+1}^{(s_c,d)}$ denotes the predicted feature for dimension d at timestep $t+1$ using solver s_c. This formulation enables per-cluster solver selection via a one-time probing pass, ensuring that HyCa combines efficiency with the adaptability of hybrid solvers.

4 Experiments

4.1 Experiment Settings

Model Configurations. We conduct experiments on four representative diffusion-based models: the text-to-image models FLUX.1-dev (Labs, 2024) and Qwen-Image (Liu et al., 2023b), the text-to-video model HunyuanVideo (Sun et al.), and the image editing model Qwen-Image-Edit (Liu et al., 2023b). To further assess compatibility with model compression techniques, we also evaluate our method on distilled models: FLUX.1-schnell and Qwen-Image-Lightning. All models are evaluated under official or recommended configurations on standard public checkpoints.

Evaluation and Metrics. For text-to-image generation, we follow the DrawBench (Saharia et al., 2022) protocol and evaluate all models on a fixed set of 200 prompts. We evaluate images using ImageReward (Xu et al., 2023) for photorealism, CLIP Score (Hessel et al., 2021) for text–image alignment, and PSNR, SSIM, LPIPS for fidelity. For text-to-video generation, we evaluate HunyuanVideo on VBench (Huang et al., 2023), which provides multi-dimensional human-aligned assessments of motion quality, visual appearance, and semantic consistency. For image editing tasks, we use GEdit-Bench (Liu et al., 2025b) to evaluate model performance across a diverse set of edit types and prompts. Unless otherwise specified, all evaluations are conducted using fixed random seeds and default inference settings. *Additional implementation details please refer to A.1.*

Table 1: **Quantitative comparison of text-to-image generation** on Qwen-Image.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	Image Reward ↑	CLIP ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Original: 50 steps	74.91	1.00×	12917.56	1.00×	1.2547 (+0.000%)	35.51	∞	1.000	0.000
50% steps	37.73	1.99×	6458.78	2.00×	1.2048 (-3.979%)	35.31	30.85	0.798	0.249
20% steps	15.31	4.89×	2583.51	5.00×	0.9234 (-26.42%)	35.17	28.52	0.627	0.504
TaylorSeer($\mathcal{N}=3$)	36.90	2.03×	4646.60	2.78×	1.0685 (-14.83%)	34.76	28.29	0.504	0.628
HyCa($\mathcal{N}=3$)	**35.33**	**2.12**×	**4646.60**	**2.78**×	**1.2363** (-1.465%)	**34.97**	**30.42**	**0.763**	**0.247**
FORA($\mathcal{N}=5$)	21.71	3.45×	2585.46	5.00×	0.7767 (-38.10%)	34.47	24.55	0.553	0.556
ToCa($\mathcal{N}=8$)	60.62	1.24×	2991.34	4.32×	0.9673 (-22.87%)	34.83	29.00	0.643	0.417
DuCa($\mathcal{N}$=9)	24.83	3.02×	2958.13	4.37×	0.8213 (-34.53%)	34.69	28.42	0.582	0.531
TaylorSeer($\mathcal{N}=6$)	24.61	3.04×	2585.46	5.00×	0.9483 (-24.41%)	34.76	28.29	0.504	0.628
HyCa($\mathcal{N}=6$)	**21.58**	**3.47**×	**2584.46**	**5.00**×	**1.1939** (-4.848%)	**34.87**	**29.65**	**0.709**	**0.320**
FORA($\mathcal{N}=6$)	17.89	4.19×	2323.30	5.56×	0.4781 (-61.91%)	28.50	28.38	0.546	0.597
ToCa($\mathcal{N}=12$)	52.72	1.42×	2406.20	5.37×	0.5593 (-55.42%)	33.92	28.72	0.589	0.519
DuCa($\mathcal{N}$=12)	21.82	3.43×	2171.56	5.95×	0.5225 (-58.34%)	33.97	28.37	0.576	0.593
TaylorSeer($\mathcal{N}=7$)	21.88	4.32×	2323.30	5.56×	0.9113 (-27.39%)	34.30	28.20	0.481	0.652
HyCa($\mathcal{N}=8$)	**13.92**	**5.38**×	**2066.81**	**6.25**×	**1.0811** (-13.84%)	**34.75**	**28.89**	**0.600**	**0.433**

Table 2: **Quantitative comparison of text-to-image generation** on FLUX.1-dev.

Method FLUX.1	Efficient Attention	Acceleration				Image Reward ↑	CLIP Score ↑
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
[dev]: 50 steps	✔	25.82	1.00×	3719.50	1.00×	0.9898 (+0.000%)	32.404 (+0.000%)
60% **steps**	✔	16.70	1.55×	2231.70	1.67×	0.9663 (-2.371%)	32.312 (-0.283%)
Δ-DiT ($\mathcal{N}=2$)	✔	17.80	1.45×	2480.01	1.50×	0.9444 (-4.594%)	32.273 (-0.404%)
Δ-DiT ($\mathcal{N}=3$)	✔	13.02	1.98×	1686.76	2.21×	0.8721 (-11.90%)	32.102 (-0.933%)
DBcache	✔	16.88	1.53×	2384.29	1.56×	1.0069 (+1.725%)	32.530 (+0.389%)
TaylorSeer ($\mathcal{N}=3, O=2$)	✔	9.89	2.61×	1320.07	2.82×	0.9989 (+0.919%)	32.413 (+0.027%)
FoCa ($\mathcal{N}=3$)	✔	9.28	2.78×	1327.21	2.80×	0.9890 (-0.081%)	32.577 (+0.533%)
HyCa ($\mathcal{N}=4$)	✔	**8.09**	**3.19**×	**967.91**	**3.84**×	**1.0182** (+2.865%)	**32.671** (+0.822%)
34% **steps**	✔	9.07	2.85×	1264.63	3.13×	0.9453 (-4.498%)	32.114 (-0.893%)
Chipmunk	✔	12.72	2.02×	1505.87	2.47×	0.9936 (+0.384%)	32.548 (+0.444%)
FORA ($\mathcal{N}=3$)	✔	10.16	2.54×	1320.07	2.82×	0.9776 (-1.232%)	32.266 (-0.425%)
ToCa ($\mathcal{N}=6$)	✘	13.16	1.96×	924.30	4.02×	0.9802 (-0.968%)	32.083 (-0.990%)
DuCa ($\mathcal{N}=5$)	✔	8.18	3.15×	978.76	3.80×	0.9955 (+0.576%)	32.241 (-0.503%)
TaylorSeer ($\mathcal{N}=4, O=2$)	✔	9.24	2.80×	967.91	3.84×	0.9857 (-0.414%)	32.413 (+0.027%)
FoCa ($\mathcal{N}=4$)	✔	9.35	2.76×	1050.70	3.54×	0.9757 (-1.424%)	32.538 (+0.414%)
Clusca ($\mathcal{N}=4, O=2, K=16$)	✔	9.25	2.79×	1045.58	3.56×	0.9850 (-0.485%)	32.441 (+0.114%)
HyCa ($\mathcal{N}=5$)	✔	**7.62**	**3.38**×	**893.54**	**4.16**×	**1.0066** (+1.700%)	**32.693** (+0.890%)
FORA ($\mathcal{N}=4$)	✔	8.12	3.14×	967.91	3.84×	0.9730 (-1.695%)	32.142 (-0.809%)
ToCa ($\mathcal{N}=8$)	✘	11.36	2.27×	784.54	4.74×	0.9451 (-4.514%)	31.993 (-1.271%)
DuCa ($\mathcal{N}=7$)	✔	**6.74**	**3.83**×	760.14	4.89×	0.9757 (-1.424%)	32.066 (-1.046%)
TeaCache ($l=0.8$)	✔	7.21	3.58×	892.35	4.17×	0.8683 (-12.28%)	31.704 (-2.159%)
TaylorSeer ($\mathcal{N}=5, O=2$)	✔	7.46	3.46×	893.54	4.16×	0.9768 (-1.314%)	32.467 (+0.194%)
FoCa ($\mathcal{N}=6$)	✔	7.54	3.42×	745.39	4.99×	0.9713 (-1.870%)	**32.922** (+1.600%)
Speca ($\mathcal{N}_{\text{max}}=8, \mathcal{N}_{\text{min}}=2$)	✔	7.42	3.48×	791.38	4.70×	0.9985 (+0.878%)	32.277 (-0.391%)
Clusca ($\mathcal{N}=5, O=1, K=16$)	✔	7.05	3.66×	897.03	4.14×	0.9718 (-1.818%)	32.319 (-0.262%)
HyCa ($\mathcal{N}=6$)	✔	6.81	3.79×	**744.81**	**5.00**×	**1.0014** (+1.163%)	32.483 (+0.244%)
FORA ($\mathcal{N}=6$)	✔	8.17	3.16×	744.80	4.99×	0.7760 (-21.62%)	31.742 (-2.043%)
ToCa ($\mathcal{N}=10$)	✘	7.93	3.25×	714.66	5.20×	0.7155 (-27.70%)	31.808 (-1.839%)
DuCa ($\mathcal{N}=9$)	✔	7.27	3.55×	690.25	5.39×	0.8382 (-15.33%)	31.759 (-1.993%)
TeaCache ($l=1$)	✔	8.19	3.19×	743.63	5.01×	0.8379 (-15.36%)	31.877 (-1.627%)
TaylorSeer ($\mathcal{N}=7, O=2$)	✔	6.77	3.81×	671.39	5.54×	0.9698 (-2.020%)	32.128 (-0.851%)
Clusca ($\mathcal{N}=6, O=1, K=16$)	✔	7.13	3.62×	748.48	4.97×	0.9704 (-1.956%)	32.217 (-0.577%)
HyCa ($\mathcal{N}=7$)	✔	**6.43**	**4.01**×	**670.44**	**5.55**×	**0.9895** (-0.030%)	**32.520** (+0.358%)

4.2 Results on Text-to-Image Generation

As shown in Table 1 HyCa achieves the best overall trade-off on **Qwen-Image** across all compression levels. At $\mathcal{N}$=3, it matches TaylorSeer in speed (2.78×) but yields higher quality (ImageReward **1.2363** vs. 1.0685, and highest PSNR **30.42**). At $\mathcal{N}$=6, HyCa remains strong (**1.1939**, **29.65**), outperforming TaylorSeer (0.9483), ToCa (0.9673), and FORA (0.7767). Even at $\mathcal{N}$=8, it sustains good visual quality (**1.0811** at 6.25×), while others drop sharply (ToCa 0.6326, FORA 0.4781). These results highlight HyCa's robustness under high acceleration while preserving visual fidelity.

On Table 2, HyCa consistently achieves the best speed–quality trade-off on **FLUX.1-dev**. At moderate acceleration ($\mathcal{N}$=5), it reaches ImageReward of **1.0066** with 4.16× FLOPs reduction, surpassing TaylorSeer (0.9857 at 3.84×) and DuCa (0.9955 at 3.80×). Even under aggressive settings, it maintains superior quality: **1.0014** at 5.00× and even **0.9895** at 5.55×, closely matching the original model (**0.9898**) while other baselines degrade (TeaCache 0.8683, ToCa 0.7155). Visual comparison on Fig. 4 further confirms its advantage in image quality under high compression.

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

Method HunyuanVideo	Efficient Attention	Acceleration				VBench ↑ Score(%)
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	
Original: 50 steps	✔	145.00	1.00×	29773.0	1.00×	80.66 (+0.0%)
22% **steps**	✔	31.87	4.55×	6550.1	4.55×	78.74 (-2.4%)
TeaCache($l=0.4$)	✔	30.49	4.76×	6550.1	4.55×	79.36 (-1.6%)
FORA($N=5$)	✔	34.39	4.22×	5960.4	5.00×	78.83 (-2.3%)
ToCa ($\mathcal{N}=5, R=90\%$)	✘	38.52	3.76×	7006.2	4.25×	78.86 (-2.2%)
DuCa ($\mathcal{N}=5, R=90\%$)	✔	31.69	4.58×	6483.2	4.48×	78.72 (-2.4%)
TaylorSeer ($\mathcal{N}=5, O=1$)	✔	34.84	4.16×	5960.4	5.00×	79.93 (-0.9%)
Speca ($\mathcal{N}_{\text{max}}=8, \mathcal{N}_{\text{min}}=2$)	✔	34.58	4.19×	5692.7	5.23×	79.98 (-0.8%)
Clusca ($\mathcal{N}=5, O=1, K=16$)	✔	35.37	4.10×	5373.0	5.54×	79.96 (-0.9%)
FoCa ($\mathcal{N}=5$)	✔	34.52	4.20×	5966.5	4.99×	79.96 (-0.8%)
HyCa ($\mathcal{N}=6$)	✔	**28.48**	**5.09**×	**5359.1**	**5.56**×	**80.25** (-0.5%)

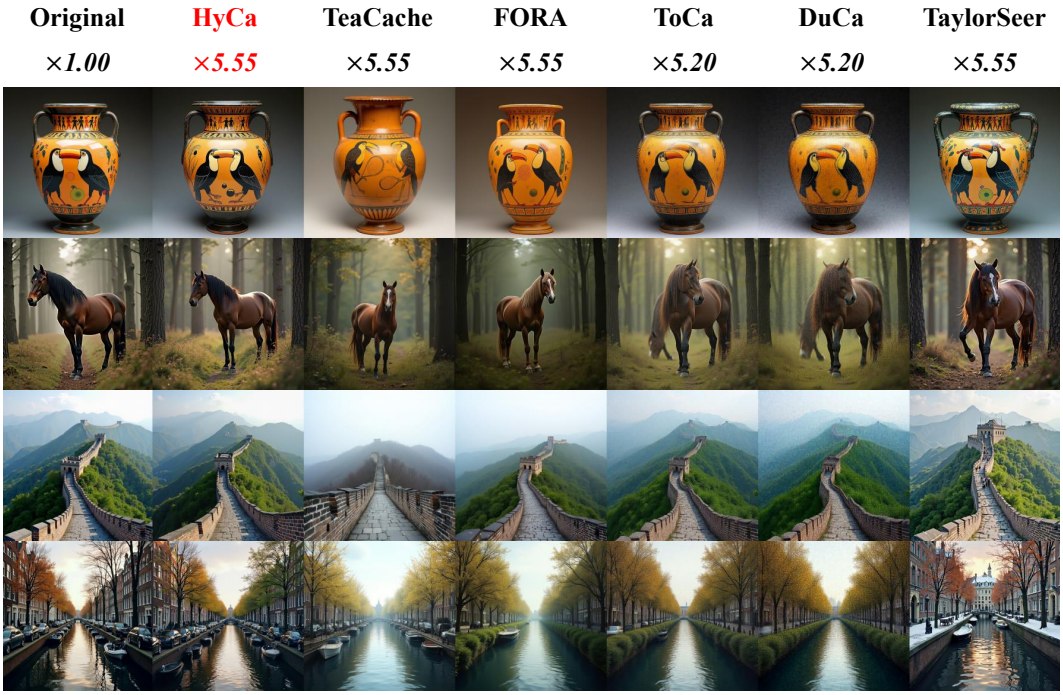

Figure 4: Visual comparison of 5.5× accelerated FLUX.

4.3 Results on Text-to-Video Generation

As shown in Table 3, our method delivers the best performance on **Hunyuan Video**. With $\mathcal{N}$=6, it achieves the highest acceleration (**5.56**× FLOPs reduction) while maintaining a strong VBench score (**80.25**), marginly lower than the original (**80.66**) at full 50-step inference. In contrast, TaylorSeer reaches only 4.16× with 79.93, TeaCache drops to 79.36, and DuCa/ToCa degrade further. This demonstrates a superior speed–quality trade-off and strong generalization to video generation.

Table 4: **Quantitative comparison of image editing** on Qwen-Image-Edit.

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($\mathcal{N}=5$)	63.15	4.51×	5643.13	5.00×	7.60	7.31	7.25	7.62	7.34	7.28
DuCa ($\mathcal{N}=6, R=90\%$)	70.95	4.01×	5897.67	4.78×	7.63	7.44	7.44	7.68	7.42	7.39
TaylorSeer($\mathcal{N}=6$)	65.66	4.33×	5643.13	5.00×	7.25	7.09	6.92	7.26	7.14	6.89
HyCa ($\mathcal{N}=6$)	**62.89**	**4.52**×	**5642.24**	**5.00**×	**7.76**	**7.49**	**7.50**	**7.77**	**7.47**	**7.45**
FORA($\mathcal{N}=7$)	52.20	5.45×	4515.74	6.24×	7.42	7.13	7.06	7.43	7.19	7.06
DuCa ($\mathcal{N}=10, R=95\%$)	59.81	4.76×	5158.45	5.46×	7.50	5.75	6.39	7.52	5.77	6.41
TaylorSeer($\mathcal{N}=8$)	53.92	5.28×	4515.74	6.24×	6.61	6.65	6.31	6.67	6.63	6.31
HyCa ($\mathcal{N}=8$)	**51.09**	**5.57**×	**4514.48**	**6.24**×	**7.74**	**7.41**	**7.44**	**7.80**	**7.36**	**7.42**

- SC denotes Semantic Consistency on Gedit Bench, PQ denotes Perceptual Quality, OS denotes the Overall Score.

4.4 Results on Image Editing

Our method also performs strongly on **Qwen-Image-Edit**, as shown in Table 4. At $\mathcal{N}$=6, it achieves the best overall scores (**7.50** CN / **7.45** EN), surpassing TaylorSeer (6.92 / 6.89), FORA (7.25 / 7.28), and even the original model's (7.41 CN). At $\mathcal{N}$=8, it remains leading (**7.44** CN / **7.42** EN) even exceeding the original model, while other baselines drop sharply (TaylorSeer 6.31 / 6.31). Visual comparisons in Fig. 5 further confirm HyCa's superiority across diverse editing tasks.

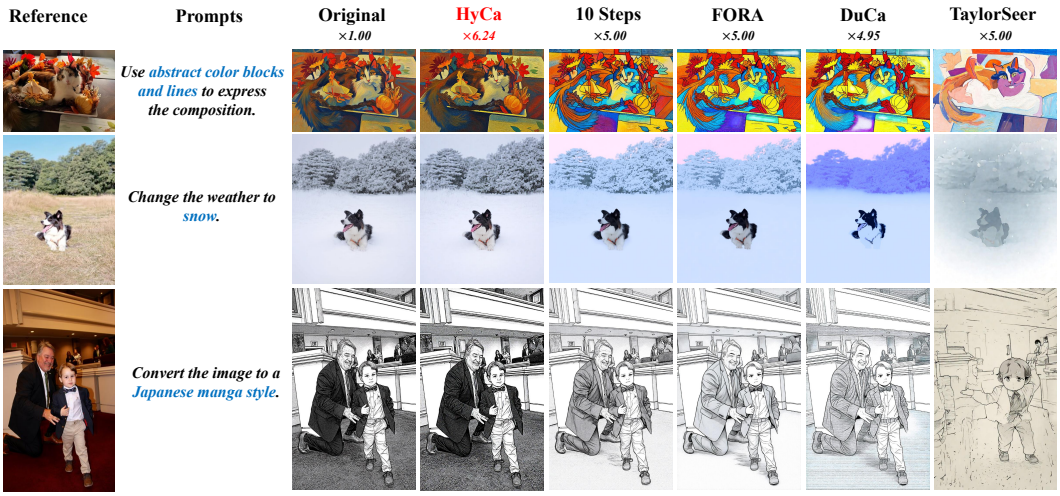

Figure 5: Visual comparison of different caching method on Qwen-Image-Edit.

4.5 Results on Distilled Models

To examine compatibility with distillation, we evaluate HyCa on **FLUX.1 schnell** and **Qwen-Image-Lightning**, as shown in Table 5. On FLUX.1-schnell, HyCa cuts latency from 2.34s to 1.16s for the 4-step distilled model (**24.4**× over the original 50-step model) while improving quality: ImageReward increased to **0.9592** and excellent perceptual metrics (PSNR 34.37, SSIM 0.928, LPIPS 0.056). On Qwen-Image-Lightning, it reduces latency from 13.35s(8-step distilled baseline) to 6.68s (**12.2**×) with quality largely preserved (ImageReward **1.2201**, CLIP 35.07) and perceptual metrics maintained (PSNR 30.97, SSIM 0.754, LPIPS 0.189). These results confirm that HyCa complements distillation, delivering extreme speedups with equal or even better quality.

Table 5: **Quantitative comparison of Distilled Model** on Flux and Qwen-Image.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	ImageReward↑	CLIP↑	PSNR↑	SSIM↑	LPIPS↓
FLUX.1[dev]: 50 steps	25.82	1.00×	3719.50	1.00×	0.9898 (+8.380%)	32.40	-	-	-
FLUX.1[schnell]: 4 steps	2.34	11.03×	297.60	12.50×	0.9133 (+0.000%)	33.85	∞	1.000	0.000
TaylorSeer ($\mathcal{N}=2$): 4 steps	1.58	16.34×	209.70	17.74×	0.9191 (+0.636%)	33.76	29.13	0.746	0.249
TeaCache ($l=0.6$): 4 steps	1.26	20.49×	163.78	22.71×	0.9023 (-1.210%)	33.87	28.01	0.379	0.734
HyCa ($\mathcal{N}=2$)**: 4 steps**	**1.16**	**22.25**×	**152.32**	**24.42**×	**0.9592** (+5.029%)	**34.18**	**34.37**	**0.928**	**0.056**
Qwen-Image: 50 steps	74.91	1.00×	12917.56	1.00×	1.2547 (+0.232%)	35.51	-	-	-
Qwen-Image-Lightning: 8 steps	13.35	5.61×	2113.67	6.11×	1.2518 (+0.000%)	35.32	∞	1.000	0.000
TaylorSeer ($\mathcal{N}=2$): 8 steps	8.11	9.24×	1320.81	9.78×	1.0418 (-16.79%)	34.44	29.49	0.62	0.377
TaylorSeer ($\mathcal{N}=3$): 8 steps	6.78	11.07×	1057.08	12.22×	0.2644 (-78.89%)	30.55	27.98	0.30	0.672
HyCa ($\mathcal{N}=2$)**: 8 steps**	8.20	9.13×	1320.81	9.78×	**1.2478** (-0.320%)	**35.27**	**32.52**	**0.837**	**0.119**
HyCa ($\mathcal{N}=3$)**: 8 steps**	**6.68**	**11.21**×	**1057.08**	**12.22**×	**1.2201** (-2.542%)	35.07	30.97	0.754	0.189

- The PSNR, SSIM, and LPIPS of HyCa are computed with respect to the outputs of the corresponding distilled baseline models.

5 Discussion

Ablation Study. We conduct our ablation study on FLUX, as shown in Fig. 7 (c)(d), HyCa consistently achieves higher ImageReward and lower prediction error than any individual solver baselines from our solver pool under the same conditions. These results confirm that HyCa benefits from combining diverse solvers rather than relying on a single integration strategy.

Why Dimension-Wise Assignment? A central design in HyCa is to assign caching strategies at feature-dimension level rather than token level primarily due to its better stability, as shown in Fig 6, clustering results in feature space remain nearly invariant once identified. Thus, solver assignments can be reused extensively, reducing both computational and data requirements. On the contrary, token-wise assignment varies with prompt and resolution, requiring frequent re-selection during inference. Empirically, Fig. 7 (a)(b) confirms that our dimension-wise assignment outperforms both token-wise (ToCa, DuCa) and one-size-fits-all (FORA, TaylorSeer) baselines.

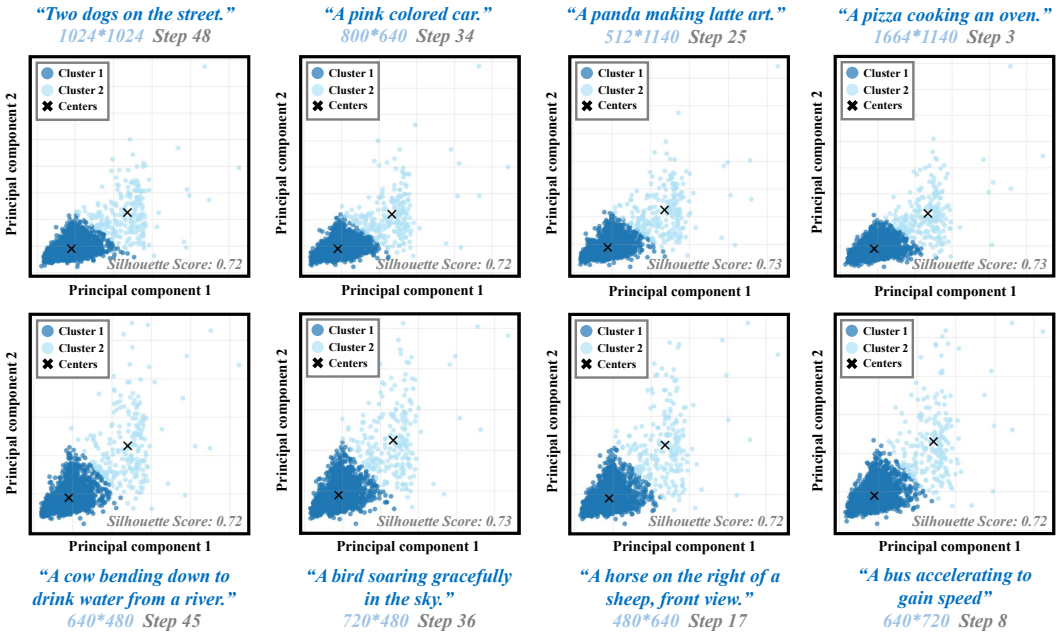

Figure 6: Top row: Clustering results from FLUX.1dev; Bottom row: Clustering results from Hunyuan Video. The clustering assignments remain highly consistent across various prompts, resolutions and timesteps, suggesting stable and robust geometric structure in the feature space.

Compatibility with Distillation. Feature caching is traditionally difficult to adapt to distilled models, as distillation drastically reduces sampling steps (e.g., from 50 to 4 or 8), making feature trajectories more discrete and oscillatory. Prior caching methods rely heavily on smooth temporal evolution and thus fail in this setting. In contrast, HyCa remains effective: its solver pool includes implicit methods suited for discrete or oscillatory dynamics, and solvers are assigned per cluster for each model. This flexibility makes HyCa fully compatible with distillation, achieving substantial acceleration while preserving generation quality.

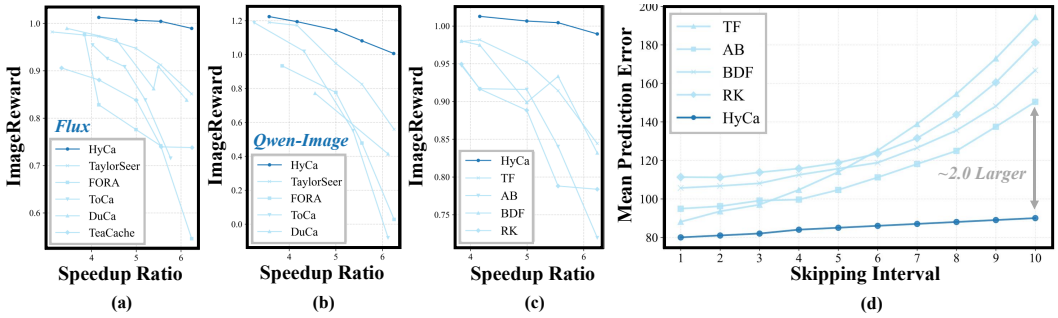

Figure 7: **Overall and ablation results of HyCa.** (a–b) HyCa consistently outperforms token-wise (ToCa) and one-size-fits-all (Taylorseer) baselines. (c–d) Ablation on FLUX shows HyCa surpasses all single-solver baselines in the solver pool, maintaining lower error and better quality.

6 Conclusion

We introduced **HyCa**, a training-free framework that reformulates feature caching in diffusion transformers as hybrid ODE solving. It clusters feature dimensions according to their temporal dynamics and assigning tailored solvers to each cluster. Our analysis shows that cluster structures in feature space are input-invariant, enabling *"One-Time Choosing"* and *"All-Time Solving"* with negligible overhead. Extensive experiments demonstrate that HyCa delivers near-lossless acceleration across text-to-image, text-to-video, and image-editing tasks, while remaining compatible with distilled models. HyCa provides a principled and versatile foundation for scaling efficient diffusion generation.

REFERENCES

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4599–4603, 2023.

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

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations*, 2024b.

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*, 2024c.

Xinle Cheng, Zhuoming Chen, and Zhihao Jia. Cat pruning: Cluster-aware token pruning for text-to-image diffusion models, 2025. URL https://arxiv.org/abs/2502.00433.

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

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.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive Benchmark Suite for Video Generative Models, November 2023. URL http://arxiv.org/abs/2311.17982. arXiv:2311.17982 [cs].

Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 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.

Sungbin Kim, Hyunwuk Lee, Wonho Cho, Mincheol Park, and Won Woo Ro. Ditto: Accelerating diffusion model via temporal value similarity. In *Proceedings of the 2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2025.

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

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*, 2023a.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17489–17499, 2023b. doi: 10.1109/ICCV51070.2023.01608.

Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems*, 36, 2024.

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.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers, 2025a. URL https://arxiv.org/abs/2503.06923.

Shiyu Liu, Yuan Han, Zhiwei Wang, Qingsong Ma, Xiaolong Li, Xianfang Zeng, Xiao Ma, Chen Zheng, Hongxin Ma, and Zhiwei Liu. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025b.

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*, 2023a.

Xuanling Liu, Yabo Huang, Jian Guo, Yong Wang, Qizhou Li, Hong Huang, Kai Luo, Yufan Shi, Lei Zhu, Jing Sun, et al. Qwen-image technical report. *arXiv preprint arXiv:2308.06642*, 2023b.

Ziming Liu, Yifan Yang, Chengruidong Zhang, Yiqi Zhang, Lili Qiu, Yang You, and Yuqing Yang. Region-adaptive sampling for diffusion transformers, 2025c. URL https://arxiv.org/abs/2502.10389.

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, 2022a.

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*, 2022b.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality, 2025. URL https://arxiv.org/abs/2410.19355.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15762–15772, 2024.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, March 2023a. URL http://arxiv.org/abs/2212.09748. arXiv:2212.09748 [cs].

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023b.

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*, pp. 234–241. Springer, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Wang, Karim Ghasemipour, Raphael Gontijo Lopes, Burcu Lee, Ekin Gontijo Lopes, Jonathan He, and et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

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

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.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1972–1981, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

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

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

Wenzhang Sun, Qirui Hou, Donglin Di, Jiahui Yang, Yongjia Ma, and Jianxun Cui. Unicp: A unified caching and pruning framework for efficient video generation, 2025. URL https://arxiv.org/abs/2502.04393.

Xingwu Sun, Yanfeng Chen, Huang, et al. Hunyuan-large: An open-source MoE model with 52 billion activated parameters by tencent. URL http://arxiv.org/abs/2411.02265.

Jiazheng Xu, Xiao Li, Guoli Xu, Yuan Zhang, Xinyu Zhang, Qishuo Zhou, Yanan Wang, Qiming Liu, Yunjie Zhang, Yuqing He, et al. Imagereward: Learning and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*, 2023.

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. URL https://openreview.net/forum?id=LQzN6TRFg9.

Zhihang Yuan, Hanling Zhang, Pu Lu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. Ditfastattn: Attention compression for diffusion transformer models. *arXiv preprint arXiv:2406.08552*, 2024a.

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*, 2024b. URL https://openreview.net/forum?id=51HQpkQy3t.

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. URL https://arxiv.org/abs/2501.00375.

Evelyn Zhang, Jiayi Tang, Xuefei Ning, and Linfeng Zhang. Training-free and hardware-friendly acceleration for diffusion models via similarity-based token pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. DPM-solver-v3: Improved diffusion ODE solver with empirical model statistics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=9fWKExmKa0.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, March 2024. URL https://github.com/hpcaitech/Open-Sora.

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.

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

Chang Zou, Evelyn Zhang, Runlin Guo, Haohang Xu, Conghui He, Xuming Hu, and Linfeng Zhang. Accelerating diffusion transformers with dual feature caching, 2024b. URL https://arxiv.org/abs/2412.18911.