Map-Guided Caching: A Global Perspective for Efficient Diffusion Transformer

Jiaqi Ji[1,†] **Ran Yang**[1] **Bo Wei**[1] **Hui Li**[1] **Hee Min Choi**[2,†]

[1]Samsung R&D Institute China Xi'an (SRCX)

[2]Samsung Electronics Co., Ltd.

{jiaqi.ji, ran01.yang, bo.wei, hui01.li, chm.choi}@samsung.com

Abstract

Feature caching is a promising way to accelerate Diffusion Transformers (DiTs) without modifying the model backbone, but existing methods largely rely on local heuristics or input-agnostic rules when deciding which intermediate features to cache and reuse. We present *Map-Guided Caching*, a feature caching framework that predicts a *Temporal Variation Map* (TVMap) for each denoising trajectory and uses the map to derive a global caching policy. The TVMap summarizes block-wise temporal variation across timesteps, while a lightweight Map Predictor estimates this variation from the prompt and initial noise before denoising starts. Given the predicted map, we formulate policy generation as a global path planning problem and combine a dynamic-programming-based orientation planner with a quantity planner that targets a desired speedup ratio. Across image generation, image editing, and video generation, the proposed method achieves up to $3.3\times$ speedup on FLUX.1 [dev] and HunyuanVideo, and $2.7\times$ on FLUX.1 Kontext [dev], while preserving competitive output quality. The advantage is especially pronounced on complex prompts, where local caching policies degrade more sharply. These results suggest that input-adaptive, globally planned caching is an effective alternative to purely local feature caching policies.

1 Introduction

Diffusion Transformers (DiTs) [37, 33] have emerged as a cornerstone architecture in modern generative modeling, enabling high-fidelity image generation, image editing, and video generation [2, 19, 21, 22]. Despite their impressive performance, the practical deployment of DiTs remains hindered by high inference costs. Specifically, DiTs require iterative denoising of latent representations across many timesteps, with each step involving the execution of a deep transformer network. Consequently, generating high-quality samples is computationally intensive and time-consuming.

Existing acceleration methods broadly follow two directions: reducing the number of denoising steps [17, 27, 30, 31, 40, 48] or reducing the cost of each step [9, 11, 24, 50]. Although effective, many such methods require distillation, fine-tuning, or structural modification of the backbone model. Such requirements introduce significant additional training overhead and data dependence.

Feature caching [23, 35] offers a different tradeoff. Instead of changing the denoising schedule or retraining the backbone model, it reuses intermediate features when neighboring timesteps produce similar outputs. Recent methods improve this idea through calibration [25, 36] or lightweight learned components [34, 15], but most policy decisions are still made either greedily or from coarse rules derived from limited observations. A critical limitation of such local strategies emerges under complex generation tasks: while short or generic prompts often induce smooth, predictable denoising

[†]Corresponding author.

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

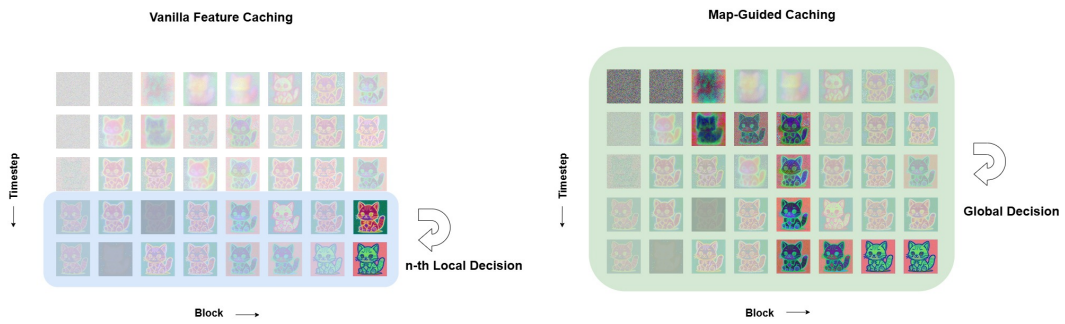

Figure 1: Comparison of local versus global caching policy generation. **(Left)** Conventional methods make greedy, stepwise reuse decisions based on immediate feature similarity. **(Right)** Map-Guided Caching leverages a predicted TVMap to plan caching globally across the denoising trajectory, enabling input-adaptive and coherent reuse strategies.

trajectories that are amenable to stepwise caching, longer and more structured prompts lead to heterogeneous temporal dynamics, where feature evolution varies significantly across timesteps and blocks. In these cases, local similarity cues become unreliable—high similarity at one step may not reflect future stability, and premature reuse can propagate errors or degrade sample quality.

This observation motivates our central question: *can feature caching be guided by a global estimate of where temporal variation will occur during denoising?* To answer the question, we introduce *Map-Guided Caching*. Our method first represents the denoising trajectory as a block-by-timestep map whose entries measure temporal feature variation. We call this representation the *Temporal Variation Map* (TVMap). A lightweight *Map Predictor* learns to estimate this TVMap from the input prompt and initial noise, effectively capturing how prompt complexity shapes the denoising dynamics. Finally, instead of making isolated reuse decisions, we derive a global caching policy from the predicted map via a path planning procedure (Figure 1), decoupling *where* to cache (spatiotemporal locations of low variation) from *how much* to cache (under a target speedup budget).

Compared to calibration-based approaches [36, 25], our method introduces a lightweight auxiliary training phase to learn an input-adaptive caching planner, enabling dynamic adaptation to diverse generation contexts. Unlike local heuristic policies [25], which make stepwise decisions based on immediate similarity, our approach expands the decision space to encompass the full denoising trajectory, allowing for globally coherent caching strategies. Empirically, this yields superior speed-quality trade-offs across multiple DiT-based generation tasks, particularly in the complex-prompt regime, where intricate temporal dynamics render local heuristics ineffective.

Our main contributions are summarized as follows:

- We introduce TVMap, a compact representation of block-wise temporal variation in DiTs, and a lightweight Map Predictor that estimates this map from the prompt and initial noise.
- We propose a global policy generation framework that combines an orientation planner and a quantity planner to derive caching policies globally from the predicted TVMap under a target speedup ratio.
- We evaluate our method on image generation, image editing, and video generation using FLUX.1 [dev] [21], FLUX.1 Kontext [dev] [22], and HunyuanVideo [19]. The approach achieves consistent acceleration while maintaining competitive output quality.

2 Related work

2.1 Diffusion transformer

Diffusion models [12, 42] replaced Generative Adversarial Networks (GANs) [8] as a dominant paradigm for generative tasks due to their impressive capabilities, and their backbone blocks also evolved from U-Net [13, 38] to transformer [44], such as DiT [37, 33]. The scalability and flexibility of DiT have enabled generation of visually convincing and context-aware content. Recent works

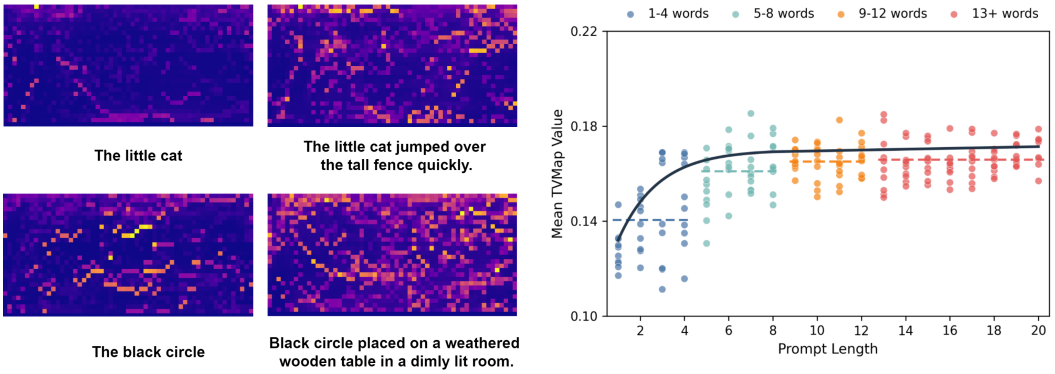

Figure 2: Relationship between prompt length and TVMap complexity. **(Left)** Qualitative comparison of TVMaps for short vs. long prompts. Longer, more descriptive prompts induce more complex patterns. **(Right)** Quantitative analysis showing average TVMap value as a function of prompt length. Each point represents one prompt; colors denote prompt-length bins containing four consecutive prompt lengths, and dashed horizontal lines indicate bin-wise averages. Shorter prompts consistently yield lower TVMap values while longer prompts exhibit significantly higher variation.

such as MMDiT [6] further extend DiT architectures to handle multimodal inputs. Despite impressive generation quality, DiT and its variants suffer from heavy computation, which limits broader applicability, especially in real-time settings.

2.2 Acceleration of diffusion transformer

To address the computational inefficiency of DiT, various acceleration methods have been proposed, broadly categorized into two directions: reducing the number of denoising steps [17, 30–32, 40, 52, 48, 27] or minimizing the computational cost per step [9, 11, 24, 47, 50, 56, 55]. The first category includes methods based on knowledge distillation [31, 32, 40, 52] and efficient solvers for ordinary differential equations (ODEs) [27, 30] and stochastic differential equations (SDEs) [17, 48]. The second category reduces per-step cost either by compressing a static model, e.g., quantization [11, 24, 47] and pruning [9, 50, 56], or by adapting the executed computation at inference time, as in dynamic neural network [10] and DyDiT++ [55]. While effective, many of these approaches require additional training or fine-tuning of the large backbone model, which is often computationally prohibitive.

2.3 Feature caching for diffusion transformer

Feature caching [23, 35] has emerged as an alternative to reduce redundant computation during the denoising process by reusing intermediate features across timesteps or transformer blocks. Recent methods such as FORA [41] and Δ-DiT [4] apply feature caching to DiT using fixed and hand-crafted policies. Meanwhile, token-wise cache methods [29, 39, 51, 58, 57] have also attracted attention, but their need to cache a large number of token-level intermediate features makes them less suitable for video generation and image editing. Learning-to-Cache [34] introduces a learnable caching scheme that is time-dependent but input-invariant. AdaCache [18] dynamically evaluates step-wise differences to determine skip lengths based on a predefined codebook. TeaCache [25] employs timestep-embedding-modulated noisy input to perform coarse estimation and a polynomial fitting procedure for refinement with 70 prompts. MagCache [36] further reduces the number of prompts required for calibration to one based on an observed magnitude law. Although calibration [25, 36] or small-scale training [34] already brings substantial savings compared with distilling or fine-tuning the backbone model, most existing feature caching methods still rely heavily on heuristics derived from limited observations and typically make caching decisions based solely on historical information.

3 Method

3.1 Preliminaries

Diffusion process. Diffusion models generate samples by reversing a learned noising process. Given an initial latent representation z_0 from the target data distribution, the forward process gradually adds Gaussian noise over T steps:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{1}$$

where $\bar{\alpha}_t = \prod_{i=1}^{t}(1 - \beta_i)$ is the cumulative noise schedule and $\beta_i \in (0, 1)$ controls the noise level at step i. After T steps, z_T becomes approximately standard Gaussian noise.

The model learns to reverse this process by estimating the noise $\epsilon_\theta(z_t, t)$ via a neural network. Starting from $z_T \sim \mathcal{N}(0, I)$, the reverse step is defined as

$$z_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(z_t, t)\right) + \sigma_t \epsilon_t, \tag{2}$$

where σ_t controls the stochasticity of the reverse process.

Feature Caching. Feature caching accelerates diffusion sampling by reusing intermediate features that change little across nearby denoising steps. Let $F_{t,i}$ denote the newly computed output of block i at step t, and let C_i store the most recent output cached for that block. The feature used during inference is

$$\widetilde{F}_{t,i} = \begin{cases} F_{t,i}, & \text{cache}, \\ C_i, & \text{reuse}. \end{cases} \tag{3}$$

Here, $\widetilde{F}_{t,i}$ is the feature passed to subsequent computation. The cache operation computes the block output and updates the cache, $C_i \leftarrow F_{t,i}$, while the reuse operation skips the block computation. Each cache is initialized by computation before its first reuse.

3.2 Temporal variation map

Understanding feature evolution throughout the denoising process is crucial for reducing redundant computation. Although intermediate feature maps contain rich information, their high dimensionality poses significant challenges for analysis. Principal Component Analysis (PCA) [1] can compress high-dimensional features into an interpretable RGB space, which provides useful visual cues, but the resulting projections are still difficult to use directly in an automated caching policy [53].

To address these limitations, we propose the TVMap, a framework that quantitatively characterizes temporal feature dynamics during the denoising process and serves as a direct guide for caching decisions. Specifically, for the feature map $F_{t,i}$ output by the i-th block at timestep t, we first apply PCA to project the original high-dimensional features into a dominant principal subspace. We then compute the feature change rate $r_{t,i}$ between consecutive timesteps using the ℓ_1 norm:

$$r_{t,i} = \frac{\|\mathrm{PCA}(F_{t,i}) - \mathrm{PCA}(F_{t+1,i})\|_1}{\|\mathrm{PCA}(F_{t,i})\|_1}. \tag{4}$$

A smaller $r_{t,i}$ indicates minimal feature variation across adjacent timesteps, implying high stability and strong potential for caching. Since the change rate cannot be computed for the final timestep due to the absence of a subsequent timestep, we assign the maximum value observed in the map as a conservative estimate for all entries in the last timestep. We aggregate the change rates across all timesteps and blocks into a matrix, which we term the TVMap.

Longer prompts induce more Complex denoising trajectories. Figure 2 presents both qualitative and quantitative evidence that prompt complexity shapes the structure of TVMap. On the left, visual comparisons of TVMaps reveal that longer, more detailed prompts produce more heterogeneous patterns. In contrast, shorter prompts yield smoother and more structured TVMaps, suggesting higher potential for caching.

To quantify this trend, the right panel plots the average TVMap value against prompt length. It shows a clear positive correlation: mean TVMap values remain low for short prompts but rise with

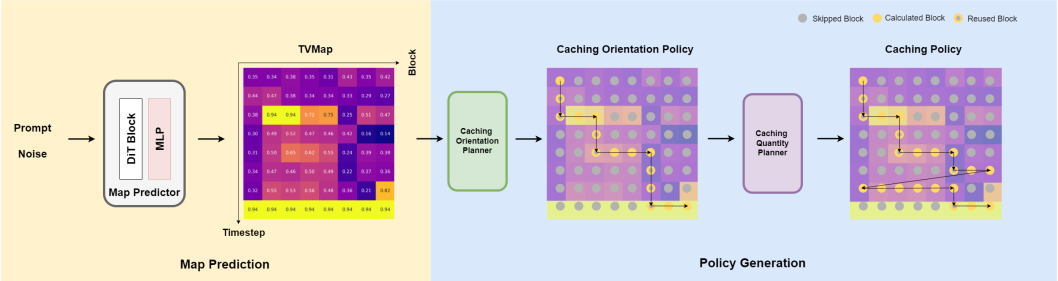

Figure 3: Overview of Map-Guided Caching. The framework first predicts a TVMap from the prompt and initial noise and then derives a global caching policy through the orientation and quantity planner.

increasing length and eventually saturate. Since the TVMap directly measures temporal feature variation, this trend indicates that denoising trajectories become increasingly irregular and harder to generalize from historical experience under complex prompts—posing greater challenges for local or static caching strategies that rely on consistent temporal redundancy.

3.3 Map-Guided caching

Map-Guided Caching comprises two stages: map prediction and policy generation, as illustrated in Figure 3. The former estimates the TVMap from the input prompt and noise, while the latter derives a global caching policy from the predicted map.

Map prediction. As defined above, TVMap captures global feature dynamics during the generation process and provides valuable cues for feature caching. However, computing the complete map requires full forward passes at every denoising step, which is infeasible during inference. To obtain a practical approximation, we design a lightweight prediction module, Map Predictor, that estimates the TVMap early in the generation process using only the initial noise and text prompt.

The Map Predictor follows an adapter design pattern and is composed of a DiT block adapted from the target model together with additional multi-layer perceptron (MLP) layers serving as an adapter. We use the first pre-trained DiT block of the target model as the feature extractor, since only the first block directly receives both the initial noisy visual patch embeddings and the full text conditioning.

The Map Predictor takes two inputs: a conditional embedding c, derived from a textual prompt via a pre-trained text encoder, and a randomly initialized latent vector x_0. It outputs a flattened TVMap $y \in \mathbb{R}^D$ defined as

$$y = F(c, x_0; \theta), \tag{5}$$

where F denotes the proposed Map Predictor parameterized by θ. The output $y = [y_1, y_2, \ldots, y_D]$ is a flattened representation of the predicted TVMap, where $D = T \times N$. Each element corresponds to a predicted change rate $\hat{r}_{t,i}$:

$$y_k = \hat{r}_{t,i}, \quad \text{where } k = t \cdot N + i. \tag{6}$$

The Map Predictor is trained in a supervised manner using mean squared error (MSE) loss between predicted and ground-truth change rates:

$$\mathcal{L} = \frac{1}{T \times N} \sum_{t=0}^{T-1} \sum_{i=0}^{N-1} (\hat{r}_{t,i} - r_{t,i})^2. \tag{7}$$

We collect 1,000 inference samples from the target model for training the Map Predictor, which takes 15.625 hours on 8 GPUs. The Map Predictor can be well trained in approximately two hours on a single NVIDIA H100 GPU with 20K training iterations. For image generation, these samples are drawn from a diversity-balanced 100-category prompt taxonomy (10 prompts per category), ensuring broad coverage of semantic and compositional complexity (see Appendix A.1 for details). This auxiliary training cost is substantially lower than retraining or distilling the full DiT backbone and is incurred only once per model setup. The sufficiency of 1,000 samples is validated in Section 4.3.

Policy generation. With TVMap, we introduce two abstractions to model the denoising process. First, regardless of whether feature caching is used, the denoising trajectory can be represented as a path on the TVMap. The start point is the top-left corner corresponding to the first block of the initial timestep, and the end point is the bottom-right corner corresponding to the final block of the last timestep. Second, there are two atomic operations: *Moving Right* and *Moving Down*. Moving Right is the standard DiT operation that advances to the next block and requires computation. Moving Down is the caching operation, signifying that the feature of the current block can be reused in the corresponding block of the next timestep. Intermediate blocks between them are skipped.

Based on these abstractions, we formulate feature caching as a path planning problem over the TVMap. Intuitively, effective caching requires balancing two objectives: reusing features in low-variation regions to save computation, and preserving explicit computation in high-variation regions to maintain generation fidelity.

To enable selection of different speedup ratios for different scenarios, we design a two-stage policy generation method consisting of a Caching Orientation Planner and a Caching Quantity Planner.

Caching Orientation Planner uses dynamic programming (DP) to determine a low-cost path:

$$P^* = \arg\min_{P \in \mathcal{A}} \sum_{(t,i) \in P} \hat{r}_{t,i}, \tag{8}$$

where P denotes a valid path through the TVMap, $\mathcal{A}$ is the set of feasible paths, and $\hat{r}_{t,i}$ serves as the node cost at block i of timestep t. The DP is solved recursively via

$$D(t,i) = \hat{r}_{t,i} + \min_{(t',i') \in \mathcal{N}(t,i)} D(t',i'), \tag{9}$$

where $D(t,i)$ represents the accumulated cost from node (t,i) to the end point, and $\mathcal{N}(t,i)$ is the set of valid predecessors under the directional constraints of moving down and moving right.

Based on the orientation policy generated in Caching Orientation Planner, the Caching Quantity Planner further refines it into a final caching policy by incorporating the target speedup ratio. Compared with directly applying a fixed threshold for allocating cache and reuse steps, we adopt a ranking-based strategy. Specifically, the change rates of all selected Moving Down operations are sorted in ascending order, and non-skipped operations are selected sequentially until the desired speedup ratio is achieved. If the Moving Down operation for block index i from timestep t to $t+1$ is selected as a non-skipped operation, then all intermediate blocks must be computed to ensure data consistency.

Extension to video generation. Video generation differs from image generation in that the caching schedule must additionally account for temporal variation. Since the Map Predictor already emits a TVMap for every frame, Map-Guided Caching extends naturally to this setting.

Concretely, we work in the latent space of the causal 3D variational autoencoder (VAE). Given a video encoded into L latent frames, the video Map Predictor outputs one TVMap per latent frame, using the same training protocol as its image counterpart. Formally, the video TVMap is a tensor $R \in \mathbb{R}^{T \times N \times L}$, whose j-th slice $R_j \in \mathbb{R}^{T \times N}$ is the TVMap of latent frame j, with $j \in \{0, \ldots, L-1\}$.

We reduce this sequence to a single TVMap using a deliberately simple rule, motivated by two properties of R. First, temporal variation is strongly non-uniform across the clip: only a few frames carry most of the motion or semantic transition, while the rest are nearly static. Second, TVMaps of adjacent latent frames are highly correlated, so neighboring frames are largely redundant and R can be subsampled along L at a fixed interval with little loss. We therefore query three key frames — the first, middle, and last, $K = \{0, \lfloor L/2 \rfloor, L-1\}$ — which capture the dominant temporal dynamics at low cost. The selected TVMaps are fused by an element-wise maximum,

$$R_{\text{final}} = \max_{j \in K} R_j \ \in \ \mathbb{R}^{T \times N}, \tag{10}$$

which yields a *conservative* caching policy: a region stays out of the cache whenever any key frame flags it as high-variation. In contrast to prior methods [25] that apply a single uniform threshold to an entire clip, R_{final} retains the spatially resolved variation patterns of individual frames. Because R_{final} matches an image TVMap in both shape and semantics, the Caching Orientation Planner and the Caching Quantity Planner apply unchanged.

Adaptive key-frame selection and learned fusion rules could plausibly improve quality further, but are beyond the scope of this work.

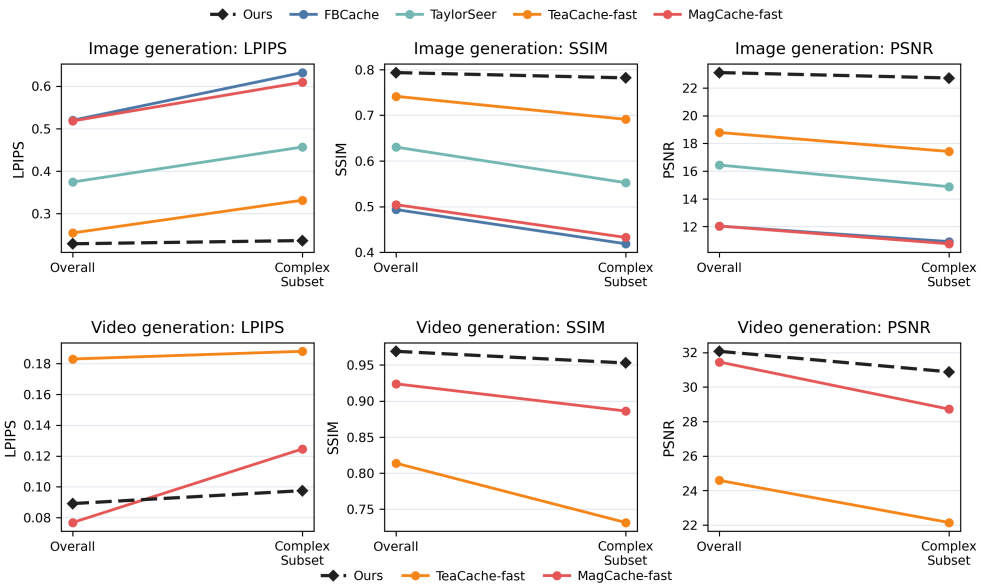

Figure 4: Performance comparison on the full dataset and its long-prompt subset for image and video generation. Competing methods exhibit significant degradation on the long-prompt subset due to high temporal dynamics, whereas our method maintains stable performance.

4 Experiments

4.1 Settings

Base models and evaluation metrics. To demonstrate the effectiveness of the proposed method, we conduct experiments across three DiT-based models covering image generation, image editing, and video generation. For image generation, we use FLUX.1 [dev] [21] to generate 1,632 high-resolution 1024×1024 images using the PartiPrompts [49] dataset with 28 timesteps. Visual quality is assessed using LPIPS [54], PSNR, and SSIM [45]. For image editing, we apply our method to FLUX.1 Kontext [dev] [22] on 606 test samples from GEdit-Bench [28], and evaluate performance with three metrics from VIEScore [20]: Semantic Consistency (SC), Perceptual Quality (PQ), and Overall Score (O). For video generation, we employ HunyuanVideo [19] to generate 1,892 videos with 129 frames at 720×1280 resolution following the VBench [16] protocol. We report both the overall VBench score and per-frame quality metrics (LPIPS, PSNR, SSIM). All speedup ratios are computed based on inference latency measured on a single NVIDIA H100 GPU.

Compared methods. We compare with recent feature caching acceleration methods, including TeaCache [25], MagCache [36], FBCache [5], TaylorSeer [26], AdaCache [18], and FORA [41]. Since our method targets high-speed regimes, we compare exclusively with the fastest and most efficient variants of prior approaches, ensuring a fair comparison at similar operating points.

Long-prompt subset. As shown in Figure 2, longer prompts indicate more complex and less smooth denoising trajectories. This implies greater feature variation across timesteps, posing significant challenges for feature caching—particularly for heuristic and fixed-pattern strategies.

To evaluate model performance under such challenging conditions, we construct a long-prompt subset for both image and video generation by selecting the top 30% longest prompts from the existing datasets. The subset focuses on complex, descriptive inputs and covers diverse semantic categories, allowing us to assess the robustness of caching mechanisms in realistic scenarios where prompts are detailed and information-rich. For open-source competitive methods, we further conduct additional experiments on this subset to highlight the performance gap under challenging prompt conditions.

Table 1: Quantitative comparison on image generation using FLUX.1 [dev] [21]. Results are reported on both the full PartiPrompts dataset [49] and its long-prompt subset.

Method	Efficiency		Overall			Long-prompt Subset		
	Speed ↑	Lat. (s) ↓	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑
FLUX.1 [dev] [21]	1.0×	14.98	–	–	–	–	–	–
FBCache [5]	2.1×	7.13	0.5198	0.4936	12.04	0.5799	0.3712	10.92
TaylorSeer [26]	2.6×	5.76	0.3743	0.6303	16.44	0.3816	0.5918	14.95
TeaCache-fast [25]	2.4×	6.24	0.2544	0.7415	18.79	0.3005	0.6439	16.69
MagCache-fast [36]	2.8×	5.33	0.5182	0.5043	12.04	0.5907	0.3754	11.07
Ours	**3.3**×	**4.53**	**0.2288**	**0.7935**	**23.11**	**0.2364**	**0.7819**	**22.72**
SCFM† [3]	3.3×	4.58	0.5731	0.3413	9.316	0.5231	0.3629	9.826

Table 2: Quantitative comparison on image editing using FLUX.1 Kontext [dev] [22].

Method	Efficiency		Visual Quality		
	Speed ↑	Lat. (s) ↓	Q_{SC} ↑	Q_{PQ} ↑	Q_O ↑
FLUX.1 Kontext [dev] [22]	1.0×	31.19	–	–	–
TeaCache-fast [25]	2.0×	15.13	6.571	7.344	6.285
MagCache-fast [36]	2.0×	15.09	6.335	**7.356**	6.083
Ours	**2.7**×	**11.56**	**6.732**	7.346	**6.453**

Table 3: Quantitative comparison on video generation using HunyuanVideo [19]. Results are reported on VBench evaluation [16] and frame-level similarity metrics. For open-source methods, we also report frame-level similarity metrics in the long-prompt subset.

Method	Efficiency		Overall				Long-prompt Subset		
	Speed ↑	Lat. (m) ↓	VBench ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑
HunyuanVideo [19]	1.0×	28.53	80.55%	–	–	–	–	–	–
AdaCache [18]	2.6×	10.97	80.25%	0.3197	0.6958	17.09	–	–	–
FORA [41]	2.8×	10.19	80.26%	0.1960	0.7688	19.45	–	–	–
TeaCache-fast [25]	2.3×	12.42	80.11%	0.1830	0.8138	24.59	0.1880	0.7316	22.15
MagCache-fast [36]	2.7×	10.57	80.37%	**0.0768**	0.9239	31.45	0.1246	0.8862	28.72
Ours	**3.3**×	**8.44**	**80.46%**	0.0891	**0.9690**	**32.96**	**0.0975**	**0.9528**	**32.07**

Table 4: Comparison of a 100-category taxonomy-based sampling strategy with random sampling under a fixed 1,000 sample budget on the image generation task using PartiPrompts [49].

Sampling method	LPIPS ↓	SSIM ↑	PSNR ↑
Random	0.2512	0.7470	19.15
100-category taxonomy	**0.2288**	**0.7935**	**23.11**

4.2 Main Results

Table 1 reports the image generation results on FLUX.1 [dev] [21]. At a high speedup ratio of 3.3×, Map-Guided Caching preserves better similarity to the full model output than TeaCache [25] and substantially improves over MagCache [36], FBCache [5], and TaylorSeer [26] on LPIPS, SSIM, and PSNR. As visualized in Figure 5, Map-Guided Caching produces outputs that are most consistent with those of the original model. Although competing methods largely preserve the semantics, they exhibit substantial deviations in composition and fine details, likely induced by denoising-trajectory drift accumulated under local caching policies. We additionally report SCFM† [3] with an 8-step sampler, a distillation-based image-generation baseline that accelerates inference by training a separate student model. This result reveals that distillation methods tend to train a student model to refit the denoising trajectory with fewer steps, which usually results in a decline in content consistency relative to the original model, even though it achieves favorable aesthetic scores and visual quality.

Table 2 reports the image editing results on FLUX.1 Kontext [dev] [22]. Our method reaches the highest speedup ratio of 2.7× while obtaining the best semantic consistency and overall score.

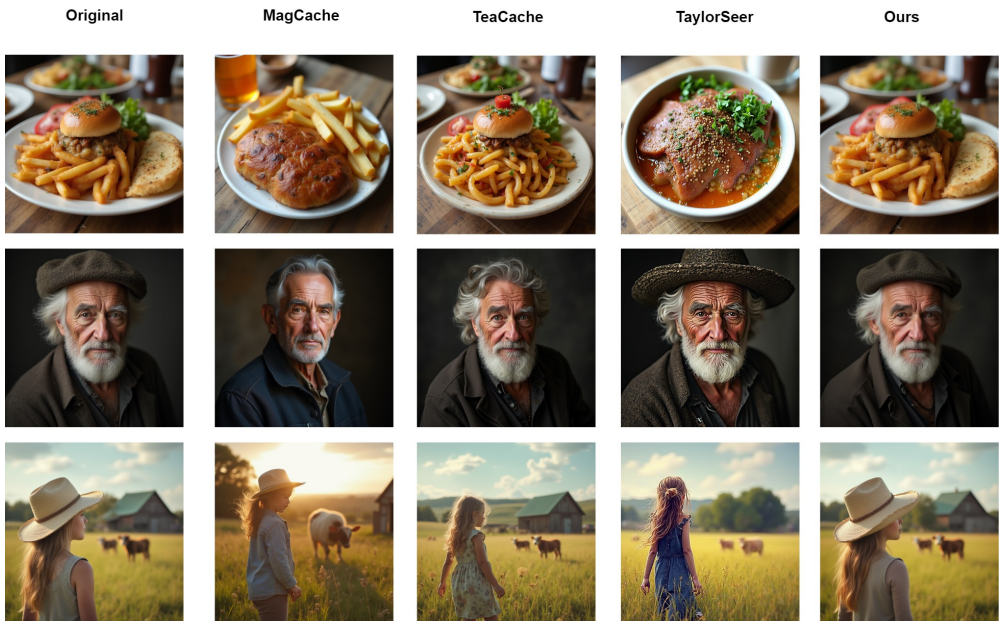

Figure 5: Visualization comparison on image generation. Map-Guided Caching maintains the highest visual fidelity to the original model's outputs, while competing methods preserve the overall composition but exhibits local discrepancies: different ingredients (row 1), a missing or altered hat (row 2), and a shifted viewpoint (row 3).

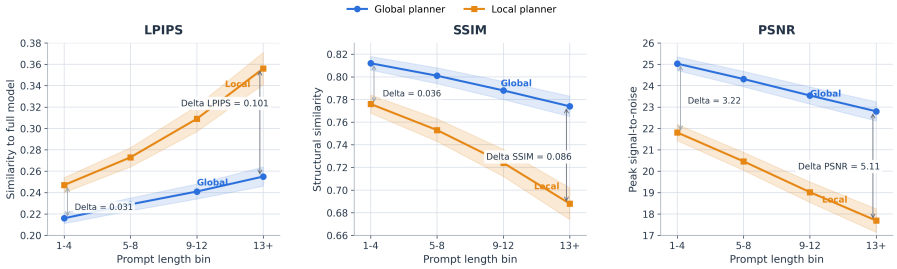

Figure 6: Complexity-stratified ablation of policy generation with FLUX.1 [dev] [21] on PartiPrompts. The proposed global planner is compared against a local variant.

Table 3 presents the quantitative results on HunyuanVideo [19]. Here the proposed method reaches 3.3× speedup while maintaining a VBench score close to the unaccelerated model and substantially improving over AdaCache [18] and FORA [41] in frame-level similarity metrics.

Qualitative comparisons on diverse tasks are provided in Appendix A.4, showing that Map-Guided Caching preserves fine details and global coherence better than competing methods under aggressive speedups. The consistent visual quality across all three generation tasks further demonstrates the generality of Map-Guided Caching.

As shown in Figure 4, Map-Guided Caching maintains stable performance on long-prompt subsets in both image and video generation, while competing methods exhibit significant degradation. This robustness underscores the advantage of input-adaptive, globally planned caching in complex generation tasks.

4.3 Ablation study

Policy Generation. Figure 6 evaluates the role of the proposed global policy generation by comparing it with a local variant that makes caching decisions based only on the final transformer block at each timestep, under varying prompt lengths. Experiments are conducted on the PartiPrompts [49] dataset using FLUX.1 [dev] [21] for image generation.

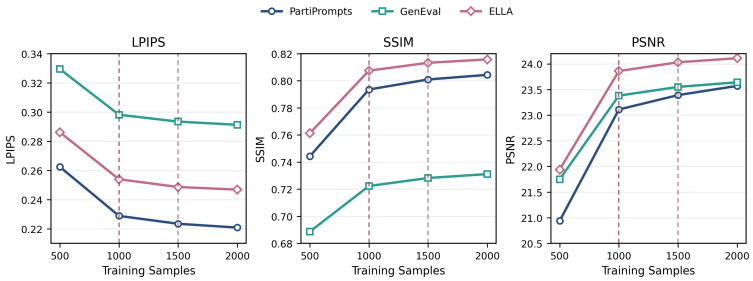

Figure 7: Ablation on training set size for the Map Predictor using FLUX.1 [dev] [21] with the 100-category taxonomy. Colors indicate different evaluation sets.

From a path-planning perspective, restricting decisions to the final block limits the policy to a TeaCache-like, stepwise strategy with a severely constrained action space. The global planner consistently outperforms the local variant across all metrics and prompt lengths. In the short-prompt regime, where denoising trajectories are smooth and feature evolution is stable, the local policy remains effective. However, as prompt length increases, temporal variation becomes more heterogeneous across blocks and timesteps (see Figure 2), rendering local decisions increasingly unreliable. Consequently, the performance gap widens, demonstrating that the global planner better captures dynamic feature dependencies and maintains robustness under complex generation tasks.

Diversity of training dataset for Map Predictor. Table 4 studies how the composition of the 1,000-sample training set affects TVMap prediction. We keep the total number of training samples fixed at 1,000 and compare a random subset against a structured 100-category taxonomy (10 prompts per category) on PartiPrompts [49]. The structured taxonomy substantially outperforms random sampling, indicating that diversity-aware coverage is more effective under the same data budget.

Size of training dataset for Map Predictor Figure 7 evaluates the impact of training set size on Map Predictor performance, with the 100-category taxonomy and balanced sampling protocol held fixed. The predictor does not relearn prompt semantics from scratch; instead, it learns a compact readout from frozen, pretrained features provided by the first DiT block—conditioned on the prompt and initial noise—to the low-dimensional TVMap target. Across PartiPrompts [49], GenEval [7], and ELLA [14], performance improves sharply from 500 to 1,000 samples, begins to saturate between 1,000 and 1,500, and plateaus by 2,000. This trend validates 1,000 samples as a practical operating point, offering a strong trade-off between auxiliary training cost and downstream generation quality, without requiring exhaustive lexical coverage or large data collection efforts.

5 Limitations

While Map-Guided Caching offers efficient and adaptive feature caching, two limitations are worth noting. First, although the auxiliary training cost for the Map Predictor is negligible compared to backbone distillation, it requires collecting and storing intermediate features from the target model during inference, resulting in a certain storage overhead. Second, our caching criterion assumes that low temporal variation implies safe feature reuse; however, even small but structured feature changes may accumulate over repeated reuse steps, potentially leading to perceptible artifacts. Future work could explore error compensation mechanisms to improve long-term stability.

6 Conclusion

We presented Map-Guided Caching, a feature caching framework that replaces heuristic local caching policies with an input-adaptive, globally planned caching strategy for DiTs. Our approach integrates a compact representation of temporal variation, a lightweight predictor, and a two-stage planner designed to achieve a target speedup while maintaining visual fidelity. Extensive experiments across image generation, image editing, and video generation demonstrate consistent acceleration with minimal impact on output quality. The performance gain is especially significant on complex prompts, where high temporal dynamics cause local caching policies to degrade rapidly. This underscores the promise of globally planned strategy as a foundational component for efficient DiT inference.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and area chair for their constructive comments. All authors are employees of Samsung Electronics Co., Ltd. and its subsidiaries, which was the sole source of funding for this work. No authors have any competing interests.

References

[1] Abdi, H. and Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459.

[2] Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., and Ramesh, A. (2024). Video generation models as world simulators. https://openai.com/research/video-generation-models-as-world-simulators.

[3] Cai, X., Wu, Y., Chen, Q., Wu, H., Xiang, L., and Wen, H. (2025). Shortcutting pre-trained flow matching diffusion models is almost free lunch. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

[4] Chen, P., Shen, M., Ye, P., Cao, J., Tu, C., Bouganis, C., Zhao, Y., and Chen, T. (2024). Δ-dit: A training-free acceleration method tailored for diffusion transformers. *CoRR*, abs/2406.01125.

[5] Cheng, Z. (2025). Paraattention: Context parallel attention that accelerates dit model inference with dynamic caching. https://github.com/chengzeyi/ParaAttention.

[6] Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. (2024). Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*.

[7] Ghosh, D., Hajishirzi, H., and Schmidt, L. (2023). Geneval: An object-focused framework for evaluating text-to-image alignment. arXiv:2310.11513.

[8] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Commun. ACM*, 63(11):139–144.

[9] Guo, J., Ouyang, W., and Xu, D. (2020). Multi-dimensional pruning: A unified framework for model compression. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1505–1514.

[10] Han, Y., Huang, G., Song, S., Yang, L., Wang, H., and Wang, Y. (2022). Dynamic neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11):7436–7456.

[11] He, Y., Liu, L., Liu, J., Wu, W., Zhou, H., and Zhuang, B. (2023). Ptqd: Accurate post-training quantization for diffusion models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 13237–13249. Curran Associates, Inc.

[12] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

[13] Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. (2022). Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646.

[14] Hu, X., Wang, R., Fang, Y., Fu, B., Cheng, P., and Yu, G. (2024). Ella: Equip diffusion models with llm for enhanced semantic alignment. arXiv:2403.05135.

[15] Huang, Y., Wang, Z., Gong, R., Liu, J., Zhang, X., Guo, J., Liu, X., and Zhang, J. (2025). Harmonica: Harmonizing training and inference for better feature caching in diffusion transformer acceleration. In *Forty-second International Conference on Machine Learning*.

[16] 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., and Liu, Z. (2024). Vbench: Comprehensive benchmark suite for video generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21807–21818. IEEE.

[17] Jolicoeur-Martineau, A., Li, K., Piché-Taillefer, R., Kachman, T., and Mitliagkas, I. (2021). Gotta go fast when generating data with score-based models. *CoRR*, abs/2105.14080.

[18] Kahatapitiya, K., Liu, H., He, S., Liu, D., Jia, M., Zhang, C., Ryoo, M. S., and Xie, T. (2024). Adaptive caching for faster video generation with diffusion transformers. arXiv:2411.02397.

[19] Kong, W., Tian, Q., Zhang, Z., Min, R., Dai, Z., Zhou, J., Xiong, J., Li, X., Wu, B., Zhang, J., Wu, K., Lin, Q., Yuan, J., Long, Y., Wang, A., Wang, A., Li, C., Huang, D., Yang, F., Tan, H., Wang, H., Song, J., Bai, J., Wu, J., Xue, J., Wang, J., Wang, K., Liu, M., Li, P., Li, S., Wang, W., Yu, W., Deng, X., Li, Y., Chen, Y., Cui, Y., Peng, Y., Yu, Z., He, Z., Xu, Z., Zhou, Z., Xu, Z., Tao, Y., Lu, Q., Liu, S., Zhou, D., Wang, H., Yang, Y., Wang, D., Liu, Y., Jiang, J., and Zhong, C. (2025). Hunyuanvideo: A systematic framework for large video generative models. arXiv:2412.03603.

[20] Ku, M., Jiang, D., Wei, C., Yue, X., and Chen, W. (2024). Viescore: Towards explainable metrics for conditional image synthesis evaluation. arXiv:2312.14867.

[21] Labs, B. F. (2024). Flux. https://github.com/black-forest-labs/flux.

[22] Labs, B. F., Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., Dockhorn, T., English, J., English, Z., Esser, P., Kulal, S., Lacey, K., Levi, Y., Li, C., Lorenz, D., Müller, J., Podell, D., Rombach, R., Saini, H., Sauer, A., and Smith, L. (2025). Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. arXiv:2506.15742.

[23] Li, S., Hu, T., Khan, F. S., Li, L., Yang, S., Wang, Y., Cheng, M.-M., and Yang, J. (2023a). Faster diffusion: Rethinking the role of unet encoder in diffusion models. arXiv:2312.09608.

[24] Li, Y., Xu, S., Cao, X., Sun, X., and Zhang, B. (2023b). Q-dm: An efficient low-bit quantized diffusion model. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 76680–76691. Curran Associates, Inc.

[25] Liu, F., Zhang, S., Wang, X., Wei, Y., Qiu, H., Zhao, Y., Zhang, Y., Ye, Q., and Wan, F. (2025a). Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 7353–7363.

[26] Liu, J., Zou, C., Lyu, Y., Chen, J., and Zhang, L. (2025b). From reusing to forecasting: Accelerating diffusion models with taylorseers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15853–15863.

[27] Liu, L., Ren, Y., Lin, Z., and Zhao, Z. (2022). Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*.

[28] 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., and Jiang, D. (2025c). Step1x-edit: A practical framework for general image editing. arXiv:2504.17761.

[29] Lou, J., Luo, W., Liu, Y., Li, B., Ding, X., Hu, W., Li, Y., and Ma, C. (2026). Token caching for diffusion transformer acceleration. arXiv:2409.18523.

[30] Lu, C., Zhou, Y., Bao, F., Chen, J., LI, C., and Zhu, J. (2022). Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5775–5787. Curran Associates, Inc.

[31] Luhman, E. and Luhman, T. (2021). Knowledge distillation in iterative generative models for improved sampling speed. arXiv:2101.02388.

[32] Luo, S., Tan, Y., Huang, L., Li, J., and Zhao, H. (2023). Latent consistency models: Synthesizing high-resolution images with few-step inference. arXiv:2310.04378.

[33] Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. (2024a). Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., and Varol, G., editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXVII*, volume 15135 of *Lecture Notes in Computer Science*, pages 23–40. Springer.

[34] Ma, X., Fang, G., Mi, M. B., and Wang, X. (2024b). Learning-to-cache: Accelerating diffusion transformer via layer caching. arXiv:2406.01733.

[35] Ma, X., Fang, G., and Wang, X. (2024c). Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15762–15772.

[36] Ma, Z., Wei, L., Wang, F., Zhang, S., and Tian, Q. (2025). Magcache: Fast video generation with magnitude-aware cache. arXiv:2506.09045.

[37] Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

[38] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

[39] Saghatchian, O., Moghadam, A. G., and Nickabadi, A. (2025). Cached adaptive token merging: Dynamic token reduction and redundant computation elimination in diffusion model. arXiv:2501.00946.

[40] Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.

[41] Selvaraju, P., Ding, T., Chen, T., Zharkov, I., and Liang, L. (2024). Fora: Fast-forward caching in diffusion transformer acceleration. arXiv:2407.01425.

[42] Song, J., Meng, C., and Ermon, S. (2021). Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[43] Sun, K., Pan, J., Ge, Y., Li, H., Duan, H., Wu, X., Zhang, R., Zhou, A., Qin, Z., Wang, Y., et al. (2023). Journeydb: A benchmark for generative image understanding. *Advances in neural information processing systems*, 36:49659–49678.

[44] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[45] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612.

[46] Wang, Z. J., Montoya, E., Munechika, D., Yang, H., Hoover, B., and Chau, D. H. (2022). DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv:2210.14896 [cs]*.

[47] Wu, J., Wang, H., Shang, Y., Shah, M., and Yan, Y. (2025). Ptq4dit: post-training quantization for diffusion transformers. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

[48] Xue, S., Yi, M., Luo, W., Zhang, S., Sun, J., Li, Z., and Ma, Z.-M. (2024). Sa-solver: Stochastic adams solver for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36.

[49] Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., and Wu, Y. (2022). Scaling autoregressive models for content-rich text-to-image generation. arXiv:2206.10789.

[50] Zhang, D., Li, S., Chen, C., Xie, Q., and Lu, H. (2025a). Laptop-diff: Layer pruning and normalized distillation for compressing diffusion models. arXiv:2404.11098.

[51] Zhang, E., Xiao, B., Tang, J., Ma, Q., Zou, C., Ning, X., Hu, X., and Zhang, L. (2024). Token pruning for caching better: 9 times acceleration on stable diffusion for free. arXiv:2501.00375.

[52] Zhang, H., Chen, X., Wang, Y., Liu, X., Wang, Y., and Qiao, Y. (2025b). Accvideo: Accelerating video diffusion model with synthetic dataset. *arXiv preprint arXiv:2503.19462*.

[53] Zhang, H., Gao, T., Shao, J., and Wu, Z. (2025c). Blockdance: Reuse structurally similar spatio-temporal features to accelerate diffusion transformers. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 12891–12900.

[54] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 586–595. Computer Vision Foundation / IEEE Computer Society.

[55] Zhao, W., Han, Y., Tang, J., Wang, K., Luo, H., Song, Y., Huang, G., Wang, F., and You, Y. (2026). Dydit++: Diffusion transformers with timestep and spatial dynamics for efficient visual generation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48(5):5755–5773.

[56] 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., Wang, B., and Barsoum, E. (2025). Dip-go: a diffusion pruner via few-step gradient optimization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

[57] Zou, C., Liu, X., Liu, T., Huang, S., and Zhang, L. (2025a). Accelerating diffusion transformers with token-wise feature caching. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R., editors, *International Conference on Learning Representations*, volume 2025, pages 75467–75488.

[58] Zou, C., Zhang, E., Guo, R., Xu, H., He, C., Hu, X., and Zhang, L. (2025b). Rethinking token-wise feature caching: Accelerating diffusion transformers with dual feature caching. arXiv:2412.18911.

A Supplementary Material

A.1 Prompt Reservoir Composition

Figure 8 visualizes the marginal composition of the diversity-balanced prompt reservoir used for predictor supervision. We build the supervision set in three stages. First, we assemble a million-scale raw prompt pool from DiffusionDB [46] and JourneyDB [43] and filter it down to a candidate reservoir of roughly 100K prompts. Second, we assign these prompts to a crossed taxonomy defined by semantic groups and structural groups, yielding a 10×10 layout with 100 prompt categories in total. Third, we perform balanced bucket-wise sampling and select 10 prompts from each category, which we define in the main paper as the *100-category taxonomy*.

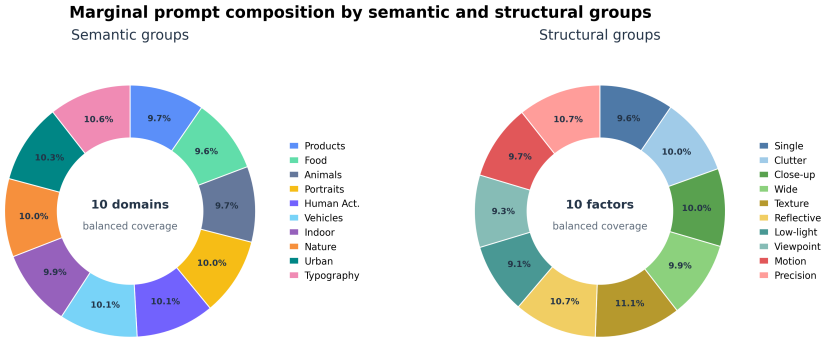

Figure 8: Marginal composition of the diversity-balanced prompt reservoir. We summarize the prompt pool by semantic groups and structural groups before bucket-wise balanced sampling.

A.2 Pseudo Algorithms

In this section, we present the algorithms for the Caching Orientation Planner and Caching Quantity Planner within the Policy Generation module. Algorithm 1 employs dynamic programming (DP) to identify a minimum-cost path P from $(1, 1)$ to (T, N) on the TVMap and additionally collects the source nodes of all down edges, $E_{\downarrow}$, which correspond to the candidate reuse transitions along the path. Subsequently, Algorithm 2 ranks these candidate transitions by their scores and retains the lowest-cost fraction determined by the hyperparameter *Reuse Ratio* ρ as reuse edges, while the remaining candidates are cancelled by forcing the affected intermediate blocks to be explicitly computed for data consistency. Based on this, each node on the map is assigned an action from set $L = \{\texttt{Cache}, \texttt{Reuse}, \texttt{Calculate}, \texttt{Skip}\}$, and an update mask S is produced to indicate the timesteps whose latent state must be updated. We set the Reuse Ratio to 75% for image generation, 68% for image editing, and 75% for video generation to achieve a favorable speedup while maintaining generation quality.

Table 5: Effect of Reuse Ratio. Reducing the Reuse Ratio implies a higher FLOPs, while leading to improved generation quality. The FLOPs is expressed in the format: denoising FLOPs + Map Predictor FLOPs.

Reuse ratio	FLOPs (T) ↓	LPIPS ↓	SSIM ↑	PSNR ↑
75%	465 + 1.05	0.2288	0.7935	23.11
50%	851 + 1.05	0.0813	0.8916	28.47
25%	1244 + 1.05	0.0215	0.9635	34.83

A.3 Additional Experiment

Effect of Reuse Ratio. We further study the quality-speed tradeoff under different reuse ratios. Lower reuse ratios require more explicit computation and therefore improve quality at the cost of

Algorithm 1 Caching Orientation Planner

Input: Predicted TVMap scores $R[1..T][1..N]$, $T, N \geq 1$
Output: Ordered path P from $(1,1)$ to (T,N); candidate reuse transitions $E_{\downarrow}$

Initialize $D[1..T][1..N] \leftarrow +\infty$, $\texttt{parent}[1..T][1..N] \leftarrow \perp$
$D[1][1] \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  for $j = 1, \ldots, N$ **do**
    if $(t, j) = (1, 1)$ **then**
      continue
    end if
    $h \leftarrow D[t][j-1]$ if $j > 1$, otherwise $+\infty$
    $v \leftarrow D[t-1][j] + R[t-1][j]$ if $t > 1$, otherwise $+\infty$
    if $h \leq v$ **then**
      $D[t][j] \leftarrow h$; $\texttt{parent}[t][j] \leftarrow (t, j-1)$
    else
      $D[t][j] \leftarrow v$; $\texttt{parent}[t][j] \leftarrow (t-1, j)$
    end if
  end for
end for
$u \leftarrow (T, N)$; $P \leftarrow [u]$; $E_{\downarrow} \leftarrow \varnothing$
while $u \neq (1, 1)$ **do**
  $p \leftarrow \texttt{parent}[u]$
  if $p = (t, j)$ and $u = (t+1, j)$ for some t, j **then**
    $E_{\downarrow} \leftarrow E_{\downarrow} \cup \{(t, j)\}$
  end if
  Prepend p to P; $u \leftarrow p$
end while
return $P, E_{\downarrow}$

Algorithm 2 Caching Quantity Planner

Input: $P, E_{\downarrow}$ from Algorithm 1, scores R, reuse ratio $\rho \in [0, 1]$
Output: $\texttt{Label}[1..T][1..N] \in \{\texttt{Cache}, \texttt{Reuse}, \texttt{Calculate}, \texttt{Skip}\}$;
update mask $S[1..T-1]$

Sort $E_{\downarrow}$ ascending by $(R[t][j], t, j)$ into list A
$K \leftarrow \lfloor \rho\, |A| \rfloor$
$U \leftarrow$ set of the first K entries of A ▷ Retained reuse edges
$B \leftarrow E_{\downarrow} \setminus U$ ▷ Cancelled reuse edges
Initialize $\texttt{Compute}[1..T][1..N] \leftarrow \texttt{False}$, $S[1..T-1] \leftarrow \texttt{False}$
$\texttt{Compute}[1][1] \leftarrow \texttt{True}$
for each right edge $((t, j-1), (t, j))$ of P **do**
  $\texttt{Compute}[t][j] \leftarrow \texttt{True}$
end for
for all $(t, j) \in B$ **do**
  $\texttt{Compute}[t][j+1..N] \leftarrow \texttt{True}$
  $\texttt{Compute}[t+1][1..j] \leftarrow \texttt{True}$
  $S[t] \leftarrow \texttt{True}$
end for
for $t = 1, \ldots, T$ **do**
  for $j = 1, \ldots, N$ **do**
    if $t > 1$ and $(t-1, j) \in U$ **then**
      $\texttt{Label}[t][j] \leftarrow \texttt{Reuse}$
    else if $\texttt{Compute}[t][j]$ **then**
      if $(t, j) \in U$ **then**
        $\texttt{Label}[t][j] \leftarrow \texttt{Cache}$
      else
        $\texttt{Label}[t][j] \leftarrow \texttt{Calculate}$
      end if
    else
      $\texttt{Label}[t][j] \leftarrow \texttt{Skip}$
    end if
  end for
end for
return $\texttt{Label}, S$

more FLOPs. For FLUX.1 [dev] [21], the Map Predictor contributes approximately 1.05T FLOPs, which is only around 0.23% of total FLOPs under the high-reuse setting.

A.4 More Samples

We provide additional qualitative comparisons for all three tasks below.



a paranoid android freaking out and jumping into the air because it is surrounded by colorful Easter eggs



A smiling sloth wearing a bowtie and holding a quarterstaff and a big book. A shiny VW van parked on grass.



a cartoon of a dog saying "I see what you did there"



the eyes of an owl

Figure 9: Image generation task. Evaluated on PartiPrompts [49], our method achieves a $3.3\times$ speedup over FLUX.1 [dev] [21] while maintaining visual quality close to the unaccelerated model.



Figure 10: Image editing task. Our method achieves a 2.7× speedup over FLUX.1 Kontext [dev] [22] while preserving strong instruction-following behavior.

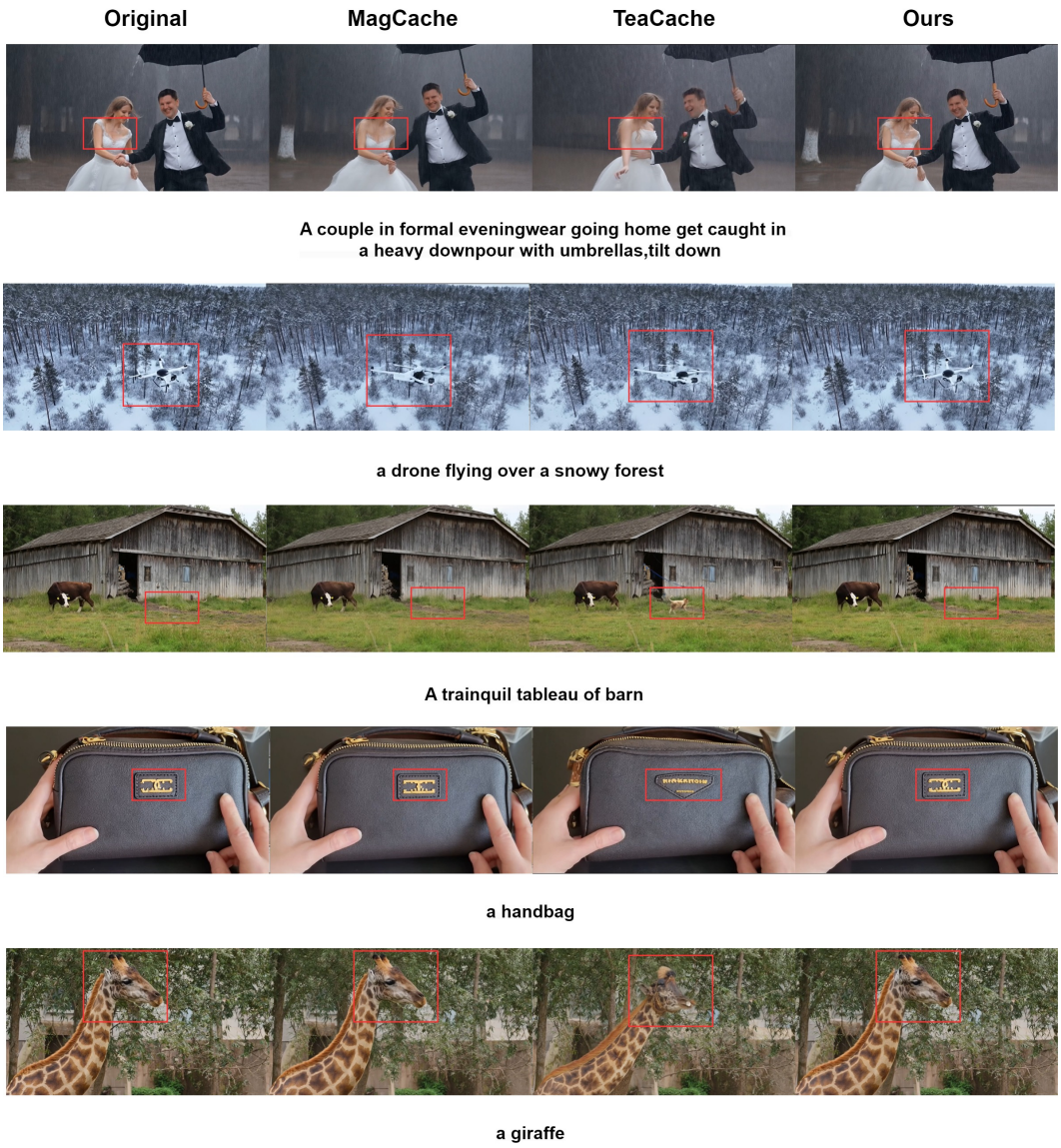

Figure 11: Video generation task. Our method achieves a 3.3× speedup over HunyuanVideo [19] while producing fewer motion artifacts than prior caching baselines.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract, introduction, and conclusion consistently describe the paper as a lightweight predictor plus global planning framework for feature caching; the main empirical claims are supported by the quantitative results and ablations in Sections 4.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes an explicit Limitations section that discusses auxiliary training and accumulated error.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The paper formalizes the global planning objective under a fixed predicted TVMap, and the supplementary material states and proves two propositions: the DP orientation planner is globally optimal for the additive TVMap cost, and the restricted local last-block planner optimizes over a subset of the global search space.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The method, predictor objective, planning procedure, evaluation models, datasets, speed targets, and auxiliary training setup are described in Sections 3 and 4, and the supplementary material further documents the prompt reservoir construction.

Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: The submission describes the experimental setup in detail, but the anonymized release of code and processed predictor-supervision assets is not included at submission time.

   Guidelines:

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

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen)?

   Answer: [Yes]

   Justification: Sections 3 and 4 specify the training iterations, supervision size, target models, datasets, evaluation metrics, and speedup settings used throughout the paper.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The submission emphasizes deterministic full-benchmark evaluations and trend analyses rather than repeated-run confidence intervals; variance is visualized for the complexity-stratified planning analysis, but not all tables include error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper reports the use of NVIDIA H100 GPUs for evaluation and predictor training and states the approximate two-hour auxiliary training cost for 20K iterations on a single H100 GPU.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research focuses on inference acceleration for existing generative backbones, does not involve deceptive reporting, and follows standard anonymous submission practices.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [No]

Justification: The current draft does not include a separate broader-impact discussion; the work is primarily a systems and efficiency contribution for generative model inference.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not introduce or release a new generative backbone, new dataset, or new public model checkpoint; it studies acceleration on existing models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper cites and credits the existing models, datasets, and baselines used in the experiments; asset provenance is documented through the references and experimental setup.

Guidelines:

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

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not claim the release of a new standalone dataset or benchmark asset as part of the submission.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or experiments with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: The work does not involve human subjects research.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are not a component of the proposed method, experiments, or analysis pipeline.

Guidelines:

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