

TAP: A Token-Adaptive Predictor Framework for Training-Free Diffusion Acceleration

Haowei Zhu^{1,2}, Tingxuan Huang¹, Xing Wang², Tianyu Zhao², Jiexi Wang², Weifeng Chen²,
 Xurui Peng², Fangmin Chen², Junhai Yong¹, Bin Wang^{1*}
¹ Tsinghua University ² ByteDance

Abstract

*Diffusion models achieve strong generative performance but remain slow at inference due to the need for repeated full-model denoising passes. We present **Token-Adaptive Predictor** (TAP), a training-free, probe-driven framework that adaptively selects a predictor for each token at every sampling step. TAP uses a single full evaluation of the model’s first layer as a low-cost probe to compute proxy losses for a compact family of candidate predictors (instantiated primarily with Taylor expansions of varying order and horizon), then assigns each token the predictor with the smallest proxy error. This per-token “probe-then-select” strategy exploits heterogeneous temporal dynamics, requires no additional training, and is compatible with various predictor designs. TAP incurs negligible overhead while enabling large speedups with little or no perceptual quality loss. Extensive experiments across multiple diffusion architectures and generation tasks show that TAP substantially improves the accuracy–efficiency frontier compared to fixed global predictors and caching-only baselines.*

1. Introduction

Diffusion Models (DMs) [8] have become a cornerstone of generative modeling, delivering state-of-the-art results in image synthesis [28] and video generation [1]. Advances in scalable architectures such as Diffusion Transformers (DiT) [27] have further improved fidelity, but the sequential nature of diffusion sampling remains a fundamental bottleneck: every denoising step requires a full forward pass through an often large and computationally expensive model. This per-step full-model requirement creates a stark trade-off between generation quality and inference efficiency that grows more severe as models and datasets scale. Existing reductions in sampling steps (e.g., DDIM [35]) mitigate some temporal dependency but cannot

fully avoid trajectory truncation errors. Complementary approaches aim to reduce per-step cost by reusing or approximating intermediate features across timesteps, yet they face their own limitations.

Many recent caching and prediction-based accelerators exploit temporal redundancy in features across adjacent timesteps [4, 32, 45]. TaylorSeer [19] formalizes temporal forecasting with Taylor-style expansions to predict multi-step feature evolution and achieves substantial savings at moderate acceleration ratios. However, prior methods typically apply a single, *global* prediction policy to all tokens and timesteps, overlooking an important empirical fact: the appropriate approximation complexity varies across tokens and over time. Low-order Taylor approximations are often sufficient for tokens that evolve slowly, but tokens that experience large displacements or rapid dynamics typically require higher-order or alternative predictors to avoid significant truncation (or extrapolation) errors. Applying a single global predictor throughout the sampling process for every sample therefore risks error accumulation and severe quality degradation at aggressive acceleration ratios. Furthermore, adaptively selecting the appropriate predictor for each token remains an open and nontrivial challenge.

To address these limitations, we propose *Token-Adaptive Predictor* (TAP), a token-adaptive diffusion acceleration framework that dynamically selects, for every token at each sampling step, the most suitable predictor from a compact family of candidates. TAP is founded on a simple, low-overhead insight: input perturbation to the model and the resulting output error are highly correlated, so a single full evaluation of the model’s first layer can serve as an informative probe of a token’s susceptibility to prediction error. Using this probe, TAP computes proxy losses for multiple candidate predictors in parallel and assigns each token the predictor with the smallest proxy error. This “probe-then-select” strategy resembles model-ensemble [29] thinking in that it leverages the complementary strengths of different predictors to assign the best predictor per token. Note that it is agnostic to the internal design of the candidate predictors and therefore can integrate a heterogeneous set of methods.

*Corresponding author: wangbins@tsinghua.edu.cn

In this work we instantiate the candidate family primarily with Taylor predictors obtained by varying expansion order and prediction horizon (distance from the expansion point). Different tokens often achieve their best predictions at different orders or in different local neighborhoods of the current timestep, so this design provides a simple yet expressive predictor pool. The per-token probe-then-select paradigm delivers three key advantages. First, it exploits token-level heterogeneity in temporal dynamics to allocate the most appropriate predictor to each token, improving approximation fidelity with negligible extra computation cost. Second, unlike prior adaptive schemes [18, 22] that require manually tuned thresholds, TAP makes decisions based on the *relative* proxy errors between predictors and thus requires no extra hand-designed thresholds. Third, the framework is architecturally flexible: the candidate set can include not only classical Taylor expansions of varying order and horizon, but also other prediction-style methods (e.g., FoCa [43], FreqCa [20]) or numerically stable polynomial regressors. Importantly, the added prediction overhead is small, predictors reduce to simple pointwise operations on cached feature tensors, and multiple predictors only require a single first-layer evaluation to validate in parallel. Specifically, compared to FLUX.1-dev with 50 steps, TAP introduces only an additional 0.1 GB of GPU memory (approximately 0.3% of the original model) while achieving an acceleration of $6.24\times$ without loss in perceptual quality.

We evaluate TAP across multiple diffusion architectures and generation tasks and show that per-token adaptive prediction substantially improves the accuracy–efficiency frontier relative to fixed global predictors and naive caching baselines. In summary, our contributions are:

- **Token-adaptive prediction framework.** We introduce TAP, a probe-driven per-token predictor selection framework that adaptively assigns the optimal predictor from a set of candidates. We show that a single full evaluation of the model’s first layer provides an effective proxy for predictor selection.
- **Taylor predictor family.** We identify that superior predictions arise at different Taylor orders and prediction horizons, and propose a family of Taylor-based predictors that together cover a diverse range of token dynamics.
- **Comprehensive evaluation.** Extensive experiments across diffusion models and tasks demonstrate that TAP consistently improves the accuracy–efficiency trade-off compared to fixed predictors and existing baselines.

2. Related Works

Diffusion models [9, 34] have achieved remarkable success in high-fidelity image and video generation, but their sequential denoising process imposes substantial computational costs. Efforts to accelerate diffusion inference broadly fall into two categories: reducing the number of

sampling timesteps, and reducing per-step computation via model-level or feature-level optimizations. Below we review representative works in these directions and situate our method within this landscape.

2.1. Sampling Timestep Reduction

A major research direction aims to reduce the number of denoising iterations while preserving sample quality. DDIM [35] introduced deterministic samplers to compress sampling trajectories, and the DPM-Solver family [23, 24, 42] leverages higher-order ODE solvers to decrease the required steps with controlled integration error. While these methods are effective at lowering step counts, they typically trade off fidelity when pushed to extreme speedups and do not directly address the high per-step cost of modern, large denoising networks. Orthogonal to solver-based acceleration, knowledge distillation techniques [26, 31, 39] compress the sampling process into far fewer network evaluations. These approaches often recover performance comparable to the original model but require additional fine-tuning. In contrast, our method is training-free and can be applied on top of both vanilla and distilled samplers. We show that it produces substantial wall-clock speedups without degrading generation quality.

2.2. Denoising Network Acceleration

Reducing per-step cost can be achieved through either model compression or feature reuse and prediction.

Model Compression. Compression methods, including pruning [6, 44], quantization [12, 16, 33], distillation [17], and token-reduction techniques [2, 11], reduce the computational footprint of denoising networks. These methods often require retraining or finetuning and balance compute savings against expressive capacity and fidelity loss.

Feature Caching and Prediction. Feature caching leverages temporal redundancy across timesteps to avoid recomputing stable activations. Methods such as DeepCache [25], Δ -DiT [4], TeaCache [18], and ToCa [45] cache and reuse intermediate features, while DiTFastAttn [40] reduces intra-step redundancy. However, these “cache-then-reuse” approaches degrade when timestep gaps widen or feature dynamics shift. Forecasting-based methods predict future features to mitigate this issue: TaylorSeer [19] applies Taylor-style expansions for multi-step prediction, FreqCa [20] introduces frequency-aware caching with Hermite interpolation for high-frequency components, and SpeCa [22] adopts a draft-then-verify strategy based on stepwise error estimation. However, these methods either lack token-level adaptivity or rely on hand-crafted thresholds, which limit scalability and robustness.

Our Approach. TAP is inspired by forecasting-based caching and speculative execution but departs from prior

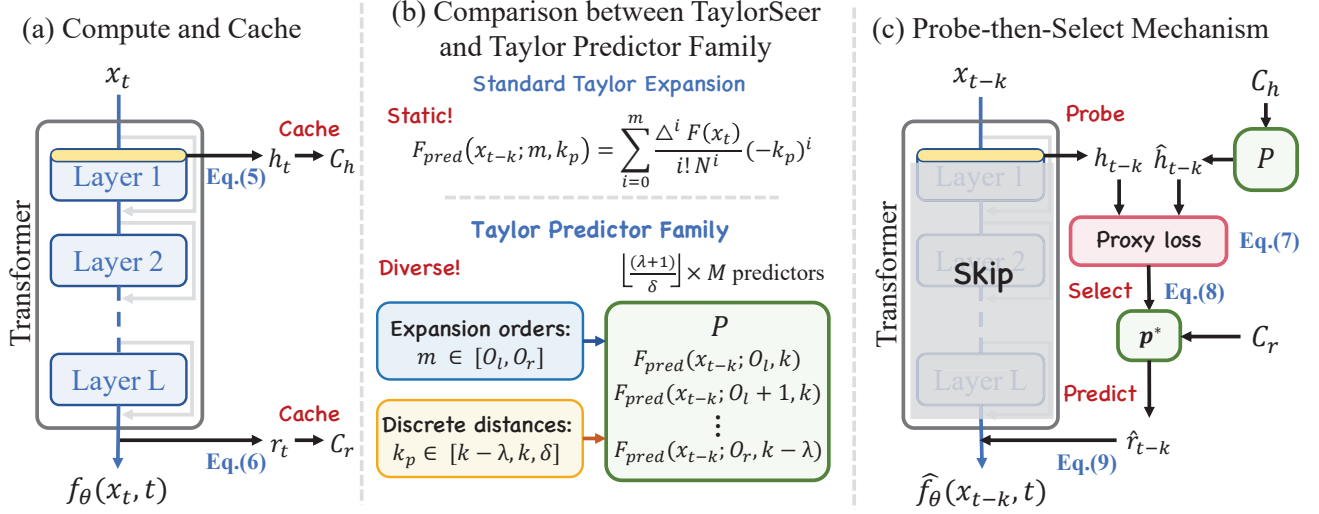


Figure 1. Overview of the TAP framework. (a) **Compute and cache**: at the first step of each N -step window TAP performs a full evaluation and caches the input–output residual and the first-layer modulated input for later probing. (b) **Taylor predictor family**: we construct a compact set of predictors by varying Taylor expansion order and prediction horizon to cover diverse token dynamics. (c) **Probe-then-select**: the cached modulated input is used as a lightweight probe to score candidate predictors per token; the selected predictor’s output replaces the full model computation for that token and step.

work in three key ways:

- (1) **Token-adaptive prediction**: rather than applying a static, global predictor to all tokens and timesteps, TAP dynamically selects predictors on a per-token basis to capture local temporal heterogeneity.
- (2) **Probe-driven scoring**: a single full evaluation of the model’s first layer serves as a lightweight probe to estimate proxy losses and score candidate predictors in parallel; unlike prior methods that use shallow-layer errors only to decide whether to execute or skip downstream computation [3, 18], TAP uses the shallow-layer probe to *select* the best predictor for each token during skipped steps.
- (3) **Threshold-free design**: decisions are based on relative proxy errors across candidates, eliminating the need for manually tuned thresholds. Together, these properties let TAP mitigate error accumulation, improve per-token prediction fidelity, and complement existing caching, forecasting, and speculative acceleration techniques.

3. Method

3.1. Preliminary

Diffusion Models. Diffusion models learn a reverse denoising process that maps Gaussian noise back to data by inverting a forward noising process. The forward step adds Gaussian noise according to a schedule $\{\alpha_t\}$:

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I), \quad (1)$$

Thus after T steps the sample is essentially noise. The learned reverse step denoises $\mathbf{x}_t$ via a neural model

$f_\theta(\mathbf{x}_t, t)$, a commonly used discrete update is:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} f_\theta(\mathbf{x}_t, t) \right) + \sigma_t \epsilon, \quad (2)$$

with $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ and σ_t controlling injected noise. Equivalently, sampling can be interpreted via reverse SDEs or ODE solvers. Because sampling involves many sequential denoising steps, each requiring a full computation of a large model, inference cost becomes a major bottleneck.

Feature Caching in Diffusion Models. Feature caching reduces per-step cost by avoiding redundant computation across timesteps. Two main strategies have been explored:

Reuse-based caching directly reuses stored activations from a previous timestep t :

$$\mathcal{F}(\mathbf{x}_{t-k}) \leftarrow \mathcal{F}(\mathbf{x}_t), \quad \forall k \in [1, N - 1], \quad (3)$$

which can skip intermediate computation and yield up to $N \times$ speedup. However, reuse assumes temporal stability. As the timestep gap k grows, approximation error accumulates rapidly and output quality degrades.

Forecast-based caching aims to predict future features rather than simply copying past ones. We formulate the forecast as a function that accepts the expansion order m and the temporal offset k as inputs:

$$\mathcal{F}_{\text{pred}}(\mathbf{x}_{t-k}; m, k_p) = \sum_{i=0}^m \frac{\Delta^i \mathcal{F}(\mathbf{x}_t)}{i! N^i} (-k_p)^i, \quad (4)$$

where $\Delta^i \mathcal{F}$ denotes the i -th finite difference (with $\Delta^0 \mathcal{F} = \mathcal{F}$ by convention), N is the temporal normalization factor and k_p denotes the prediction distance ($k_p = k$ in [21]). Forecasting is more accurate than direct reuse for modest k , but it relies on local smoothness and well-behaved higher-order differences. In practice, truncation and numerical instability cause prediction error to grow quickly with k , especially under noisy or complex dynamics.

Moreover, both reuse and global-forecast approaches typically apply the same strategy uniformly across all tokens. This ignores a key empirical fact: *tokens differ in temporal sensitivity*. Some tokens (e.g., smooth background regions) remain stable and are well-served by low-cost reuse or low-order predictors, while others (e.g., edges, moving objects) require higher-order or alternative predictors. Long-range mistakes on a small set of sensitive tokens can disproportionately degrade overall generation quality. These observations motivate token-adaptive, cost-aware prediction schemes that select prediction complexity per token and per step.

3.2. Overview of TAP

To overcome the limitations of prior diffusion accelerators that apply a fixed global predictor at every sampling step, we propose the **Token-Adaptive Predictor** (TAP), a probe-driven framework that makes *per-token* predictor choices. As illustrated in Figure 1, TAP dynamically assigns the predictor with the smallest proxy error to each token at every step, reducing overall error by exploiting token-level heterogeneity in temporal dynamics and validating candidates with a lightweight first-layer probe. Below we summarize TAP’s core components: how we compute and cache compact proxies, the construction of a diversified Taylor predictor family, and the probe-then-select mechanism used to pick predictors per token and step.

Let $\mathbf{r}_t \in \mathbb{R}^{B \times N_x \times D}$ denote cached features at timestep t for a batch of B samples, each containing N_x tokens of dimension D . Our goal is to predict features at future timesteps $t - k$, where $k \in \{1, \dots, N - 1\}$, using previously cached full evaluations while avoiding expensive full forward passes for every token and step.

Compute and Cache. At the beginning of every N -step denoising window, TAP performs one full model evaluation and accelerates the following $N - 1$ steps through feature prediction. During this full step, TAP caches the modulated first-layer input and the global residual between the model’s input and output, which are later used to estimate the feature evolution. Specifically, given the input $\mathbf{x}_t$ at timestep t , the modulated first-layer input and residual are defined as:

$$\mathbf{h}_t = \text{Modulate}(\text{Norm}_1(\mathbf{x}_t), \mathbf{s}_t, \mathbf{g}_t), \quad (5)$$

$$\mathbf{r}_t = f_\theta(\mathbf{x}_t, t) - \mathbf{x}_t. \quad (6)$$

where $\mathbf{s}_t$ and $\mathbf{g}_t$ denote the learned shift and scale parameters at timestep t , $\text{Norm}_1(\cdot)$ is the first-layer normalization, and f_θ represents the denoising network. These cached tensors serve as efficient proxies for subsequent probe-based scoring and prediction without requiring full downstream propagation.

TAP then leverages these cached features to perform token-level forecasting using a compact set of candidate predictors, selecting the optimal one for each token based on probe-computed proxy losses. This mechanism enables high-accuracy, training-free acceleration while maintaining generation fidelity.

Taylor Predictor Family. During the reverse denoising process of diffusion models, hidden features evolve smoothly along the sampling trajectory, which enables local prediction based on previous features, as shown in Eq.(4). However, a single predictor cannot fit all tokens equally well. Recent studies such as FreqCa [20] highlight that low-frequency and high-frequency tokens exhibit distinct temporal dynamics, thus requiring different predictors. Besides, ToCa [45] selectively reuses only tokens with small important scores.

To capture this token-level heterogeneity, we construct a **predictor family** with varying levels of complexity. Specifically, we design M Taylor expansion predictors of different orders by adjusting $m \in [O_l, O_r]$ in Eq.(4). Low-order predictors are more robust when feature evolution is abrupt or discontinuous, while high-order predictors better approximate smooth and stable dynamics.

Moreover, different tokens may exhibit different convergence ranges for Taylor expansions. If the prediction horizon k_p in Eq.(4) exceeds a token’s convergence radius, Taylor-based predictions, especially high-order ones, can diverge and amplify accumulated error, leading to unstable forecasts. To mitigate this, we further diversify predictors by varying the prediction distance k_p from $k - \lambda$ to k (where k is the number of steps since the last full computation), and discretize this range with an interval δ .

In total, our Taylor predictor family contains $\left\lfloor \frac{(\lambda+1)}{\delta} \right\rfloor \times M$ candidate predictors, where λ and δ control the discretized prediction-horizon range and step, and M control the available Taylor orders (we include orders 0, 1, 2). By default we use $M = 3$, $\lambda = 4$, and $\delta = 1$, yielding $\lfloor (4+1)/1 \rfloor \times 3 = 15$ predictors. In practice TAP is not sensitive to these hyperparameters: a compact, diversified pool suffices for strong, consistent gains across models and benchmarks, as shown empirically in Section 4.

Probe-then-Select Mechanism. After constructing a set of candidate predictors, an open challenge is how to robustly and efficiently combine them at inference time. Re-

cent work shows that shallow-layer activations can serve as an effective cache indicator [5, 18]. Following this insight, we use the modulated first-layer input $\mathbf{h}_t$ (Eq. (5)) as a proxy for predictor quality: a predictor whose *predicted* modulated input is closer to the *actual* modulated input is likely to yield a more accurate downstream prediction.

Concretely, let $\mathcal{P}$ denote the predictor family and let $\hat{\mathbf{h}}_{t,p} \in \mathbb{R}^{B \times N_x \times D}$ be the modulated-input predicted by predictor $p \in \mathcal{P}$ for the current step at time t . We compute a lightweight, per-token proxy error between each predictor’s prediction and the actual activated modulated input. For token index (b, n) the proxy loss for predictor p is

$$\mathcal{L}_p^{b,n} = d(\hat{\mathbf{h}}_{t,p}^{b,n}, \mathbf{h}_t^{b,n}), \quad (7)$$

where $d(\cdot, \cdot)$ denotes a chosen distance metric (e.g., L1, cosine, or MSE). In this work we empirically use cosine distance. Given the proxy losses, we select the best predictor for each token as follows:

$$p^{*,b,n} = \arg \min_{p \in \mathcal{P}} \mathcal{L}_p^{b,n}. \quad (8)$$

All predictions $\hat{\mathbf{h}}_{t,p}$ are produced in parallel from cached full-step features, and the losses in Eq.(7) are computed elementwise over tokens and batch entries, making the probe-and-select stage highly efficient. Once $p^{*,b,n}$ is determined, TAP applies the corresponding predictor for that token in the remainder of the network for the current step. Denoting the per-token residual predicted by predictor $p^{*,b,n}$ as $\hat{\mathbf{r}}_{t,p^{*,b,n}}^{b,n}$, the predicted per-token residuals assemble into the full predicted residual $\hat{\mathbf{r}}_t$, and the model’s final predicted output for the current step is:

$$\hat{f}_\theta(\mathbf{x}_t, t) = \mathbf{x}_t + \hat{\mathbf{r}}_t. \quad (9)$$

A pseudo-algorithm of TAP is provided in Algorithm 1. Two practical benefits of our method are as follows. First, because selection uses only *relative* proxy errors across predictors (Eq.(8)), TAP does not require any hand-tuned thresholds or calibration heuristics. Second, although predictors differ, they all derive from the same distribution of previously computed full-step features. Replacing a token’s downstream computation with a prediction from that shared distribution is compatible with the intrinsic overridability of diffusion sampling [15], which permits local substitutions in intermediate states without disrupting global inference. Together, these properties yield a scalable, threshold-free mechanism for per-token predictor selection that mitigates error accumulation while remaining training-free.

3.3. Complexity Analysis

Standard diffusion sampling requires T sequential steps each performing a full forward pass. TAP reduces this cost by introducing token-adaptive predictions and a lightweight

Algorithm 1 Pseudocode of TAP

Require: denoiser f_θ , predictor set $\mathcal{P}$, window N , distance $d(\cdot, \cdot)$

```

1:  $C_h \leftarrow \emptyset, C_r \leftarrow \emptyset$   $\triangleright$  compact cache: first-layer modulated input and residual
2: for  $t \leftarrow T$  downto 1 do
3:    $\mathbf{x}_t \leftarrow$  current model input
4:    $\mathbf{h}_t \leftarrow \text{Modulate}(\text{Norm}_1(\mathbf{x}_t), \mathbf{s}_t, \mathbf{g}_t)$   $\triangleright$  Eq.(5)
5:   if  $t \bmod N = 0$  then
6:      $\mathbf{r}_t \leftarrow f_\theta(\mathbf{x}_t, t) - \mathbf{x}_t$   $\triangleright$  full residual (Eq.(6))
7:      $C_h \leftarrow \mathbf{h}_t, C_r \leftarrow \mathbf{r}_t$   $\triangleright$  store compact proxies
8:     use  $f_\theta(\mathbf{x}_t, t)$  as the model output for this step
9:   else
10:    for all  $p \in \mathcal{P}$  do  $\triangleright$  parallel prediction from cached proxies (e.g., Taylor variants)
11:       $\hat{\mathbf{h}}_{t,p} \leftarrow \text{Predict}(p, C_h)$   $\triangleright$  (Eq.(4))
12:    end for
13:    for all tokens  $(b, n)$  do
14:       $p^{*,b,n} \leftarrow \arg \min_p d(\hat{\mathbf{h}}_{t,p}^{b,n}, \mathbf{h}_t^{b,n})$   $\triangleright$  proxy error (Eq.(7)) and selection (Eq.(8))
15:       $\hat{\mathbf{r}}_t^{b,n} \leftarrow \text{Predict}(p^{*,b,n}, C_r)$   $\triangleright$  (Eq.(4))
16:    end for
17:     $\hat{\mathbf{r}}_t \leftarrow \hat{\mathbf{r}}_t^{b,n}$   $\triangleright$  assemble per-token residuals
18:     $\hat{f}_\theta(\mathbf{x}_t, t) \leftarrow \mathbf{x}_t + \hat{\mathbf{r}}_t$   $\triangleright$  derive the predicted output (Eq.(9))
19:    use  $\hat{f}_\theta(\mathbf{x}_t, t)$  as the model output for this step
20:  end if
21: end for
```

probe-based verification instead of executing full downstream computation at every step. The computation introduced by TAP consists of one full evaluation per N -step window, parallel prediction of a compact candidate set (implemented with low-cost pointwise and small-polynomial operations), and the first-layer activation used as the probe. These prediction and verification operations are inexpensive in practice: on FLUX.1-dev with 50 steps TAP adds only 0.1 GB of GPU memory (0.3% of the original model) and incurs roughly 0.015% additional FLOPs compared to a single global-predictor baseline. Importantly, TAP stores only the per-step residual and the modulated first-layer input, so the extra memory is constant with respect to model depth (i.e., $O(1)$), whereas methods that cache every layer incur $O(L)$ memory overhead [21, 45]. Wall-clock latency gains depend on model size and hardware, we report measured latency results in the experiments.

4. Experiment

4.1. Experimental Settings

Model Configurations. We evaluate TAP on multiple state-of-the-art diffusion architectures to demonstrate broad

applicability and robustness. Specifically, we test on **FLUX.1-dev** [14], **Qwen-Image** [37] (and their few-step distilled variants), and a large-scale video generation model, **HunyuanVideo** [13]. For controlled comparisons we reimplement or adopt public checkpoints of representative acceleration baselines, including FORA [32], TeaCache (CVPR’25) [18], SpeCa (ACM MM’25) [22], and TaylorSeer (ICCV’25) [19].

Model Configurations. For text-to-image evaluation we use DrawBench [30] and report semantic alignment and perceptual quality via ImageReward [38] and CLIP Score [7]; for reconstruction-style metrics we additionally report PSNR, SSIM [36] and LPIPS [41]. Video experiments follow the VBench [10] protocol: for each of 950 prompts we generate 5 videos under different seeds (4,750 videos total) at 480p (2s duration, 17 frames at 8 FPS). Latency and FLOPs are measured on the corresponding benchmarks using NVIDIA H20 GPUs (single node). We report the average wall-clock latency and FLOPs per generated sample with a batch size of 1.

4.2. Image Generation

4.2.1. FLUX.1-dev

We evaluate TAP on FLUX.1-dev [14] across multiple acceleration regimes and report generation quality, perceptual metrics, average latency, and FLOPs (Table 1). Direct-reuse methods (FORA, TeaCache) exhibit noticeable quality degradation, and global-forecast methods (TaylorSeer, SpeCa) also degrade at long cache horizons (e.g., $N = 7$). By contrast, TAP attains a superior accuracy–efficiency frontier relative to these direct-reuse and global-forecast baselines: for instance, at $6.24\times$ acceleration, TAP preserves perceptual and fidelity metrics with no measurable quality loss while yielding large latency and FLOPs reductions. This improvement stems from TAP’s token-adaptive assignment of predictors, which preserves sample quality at matched computational budgets. On the distilled model FLUX-Schnell, we further validate the effectiveness of TAP, achieving up to $2\times$ acceleration without noticeable quality loss. Surprisingly, TAP sometimes yields slightly higher ImageReward and CLIP (e.g., 0.99 and 31.19 at $6.24\times$ vs. 0.97 and 30.86 at $4.16\times$). This may stem from TAP’s token-adaptive prediction benefiting from larger prediction horizons, which for some prompts produces cleaner or more consistent outputs.

4.2.2. Qwen-Image

On Qwen-Image [37] (Table 2), TAP yields consistent improvements in both quality preservation and latency reduction compared to TaylorSeer and other cache-based methods. At a $3.57\times$ speedup, TAP attains an ImageReward of 1.23 (vs. 1.18 for TaylorSeer). On perceptual metrics TAP

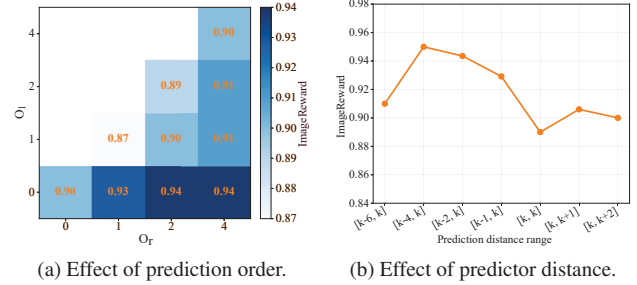


Figure 2. **Ablation on Taylor predictor family.** We investigate the influence of distance range and order configuration in TAP with $N = 7$ on FLUX.1-dev. In (a), we fix the prediction distance as k , and in (b), we set the order as 2 and $\delta = 1$.

also shows clear gains: at $3.57\times$ acceleration TAP’s PSNR exceeds TeaCache by about 1.1 dB and TaylorSeer by about 2.1 dB. Moreover, on the distilled Qwen-Image-Lightning variant TAP achieves substantial speedups with only minor quality degradation. These results indicate that TAP better preserves perceptual quality at high speedups while delivering meaningful latency reductions.

4.3. Video Generation

Table 3 demonstrates that TAP attains the best accuracy–efficiency trade-off on HunyuanVideo [13]. With a cache window of $N = 6$, TAP achieves a $4.98\times$ speedup and the highest VBench score (65.46), corresponding to only a 1.7% drop versus the unaccelerated baseline, i.e., near-lossless visual quality with substantially reduced computation. Qualitative results in Figure 4 further show that TAP better preserves high consistency and fidelity, validating the effectiveness of token-adaptive predictor assignment for balancing efficiency and output quality.

4.3.1. Ablation Study

Ablation of the Taylor Predictor Family We construct a Taylor predictor family and study how predictor count and type affect generation quality. Concretely, we vary the prediction horizon (the discretized range $[k - \lambda, k]$) and the Taylor order range $[O_l, O_r]$, reporting results in Figure 2. Our ablation shows that increasing either the prediction distance or the expansion order yields substantial gains (raising a baseline ImageReward from about 0.89 ($O = 2$) to 0.95). The best single setting in our sweep was $O_l = 0$, $O_r = 2$ with $\lambda = 4$, and jointly varying both distance and order produced the largest improvement (from 0.95 to 0.99). Performance continues to improve as the predictor family grows but with diminishing returns and eventual saturation. Increasing granularity (e.g., using $\delta = 0.1$ instead of $\delta = 1$) produces only a small additional gain (about 0.005 ImageReward), so we use $\delta = 1$ by default for simplicity.

Two additional observations emerge. First, including zeroth-order predictors (order 0) is particularly valuable:

Table 1. **Quantitative comparison in text-to-image generation** for FLUX.1-dev and FLUX.1-schnell (a distilled version). Best results are highlighted in **bold**, and second-best are underlined.

Method	Acceleration				Quality Metrics		Perceptual Metrics		
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	ImageReward↑	CLIP↑	PSNR↑	SSIM↑	LPIPS↓
50 steps	42.18	1.00×	3726.87	1.00×	0.95	30.63	-	-	-
40% steps	17.08	2.47×	1490.75	2.50×	0.93	30.84	15.60	0.69	0.40
30% steps	<u>13.14</u>	<u>3.21</u> ×	<u>1118.06</u>	<u>3.33</u> ×	<u>0.92</u>	<u>30.70</u>	<u>14.81</u>	<u>0.66</u>	<u>0.44</u>
20% steps	8.52	4.95 ×	745.37	5.00 ×	0.89	30.28	13.96	0.62	0.50
FORA ($N = 4$)	12.37	3.41×	969.93	3.84×	0.87	30.45	14.28	0.64	0.47
TeaCache ($l = 1.1$)	12.33	3.42×	913.98	4.08×	0.93	30.47	15.19	0.65	0.47
TaylorSeer ($N = 5, O = 2$)	12.02	3.51×	895.32	4.16×	0.95	30.79	<u>16.73</u>	0.71	0.36
SpeCa ($N = 2, M = 8$)	11.68	3.61 ×	845.41	4.41 ×	<u>0.96</u>	30.73	16.09	0.68	0.40
TAP ($N = 5$)	<u>11.82</u>	<u>3.57</u> ×	<u>895.08</u>	<u>4.16</u> ×	0.97	30.86	17.43	0.71	0.34
FORA ($N = 5$)	<u>10.12</u>	<u>4.17</u> ×	746.29	4.99×	0.89	30.55	13.82	0.61	0.50
TeaCache ($l = 1.4$)	11.16	3.78×	779.83	4.77×	0.92	30.44	14.79	0.63	0.51
TaylorSeer ($N = 6, O = 2$)	10.02	4.21 ×	<u>746.29</u>	<u>4.99</u> ×	<u>0.94</u>	<u>30.76</u>	<u>15.88</u>	<u>0.67</u>	<u>0.42</u>
SpeCa ($N = 4, M = 8$)	10.76	3.92×	755.32	4.93×	0.93	30.57	15.72	0.66	0.43
TAP ($N = 6$)	10.21	4.13×	746.02	5.55 ×	0.96	30.92	16.76	0.68	0.39
FORA ($N = 7$)	<u>7.91</u>	<u>5.33</u> ×	597.26	6.24×	0.80	30.42	13.43	0.60	0.55
TeaCache ($l = 2.0$)	8.59	4.91×	604.02	6.17×	0.66	30.07	14.23	0.60	0.58
TaylorSeer ($N = 8, O = 2$)	8.17	5.16×	597.26	6.24×	0.91	30.62	14.72	0.61	0.50
TAP ($N = 8$)	8.40	5.02×	<u>596.99</u>	<u>6.24</u> ×	0.99	31.19	16.11	0.64	0.44
TAP ($N = 10$)	6.21	6.79 ×	522.48	7.13 ×	<u>0.91</u>	<u>30.69</u>	<u>15.94</u>	<u>0.63</u>	<u>0.49</u>
FLUX-Schnell (4 steps)	4.46	1.00×	278.41	1.00×	0.88	32.03	-	-	-
TaylorSeer ($N = 3, O = 2$)	2.61	1.71×	139.23	2.00×	0.86	32.08	22.67	0.72	0.32
TAP ($N = 3$)	2.55	1.75 ×	139.12	2.00 ×	0.88	32.36	26.26	0.86	0.15

Table 2. **Quantitative comparison in text-to-image generation** for Qwen-Image and Qwen-Image-Lightning (a distilled version). Best results are highlighted in **bold**, and second-best are underlined.

Method	Acceleration			Quality Metrics		Perceptual Metrics		
	Latency (s) ↓	FLOPs (T) ↓	Speed ↑	ImageReward ↑	CLIP ↑	PSNR ↑	SSIM ↑	LPIPS ↓
50 steps	127.10	12917.56	-	1.23	33.74	-	-	-
50% steps	<u>63.54</u>	6458.78	2.00×	1.20	33.77	18.35	0.77	0.28
20% steps	25.54	2583.51	5.00 ×	<u>0.73</u>	<u>32.24</u>	<u>13.45</u>	<u>0.57</u>	<u>0.57</u>
FORA ($N = 3$)	48.59	4392.93	2.94×	0.92	32.25	14.12	0.59	0.50
TeaCache ($l = 0.8$)	52.64	3621.18	3.57×	<u>1.18</u>	<u>33.52</u>	<u>19.07</u>	<u>0.79</u>	<u>0.27</u>
TaylorSeer ($N = 4, O = 2$)	<u>44.37</u>	<u>3617.96</u>	<u>3.57</u> ×	1.18	33.44	18.02	0.76	0.30
TAP ($N = 4$)	40.96	3617.43	3.57 ×	1.23	33.80	20.13	0.81	0.22
FORA ($N = 4$)	<u>38.41</u>	3359.99	3.84×	0.70	31.98	13.13	0.53	0.58
TeaCache ($l = 1.0$)	47.84	3276.45	3.94×	<u>1.16</u>	<u>33.47</u>	<u>18.13</u>	<u>0.73</u>	<u>0.34</u>
TaylorSeer ($N = 5, O = 2$)	40.70	3101.30	4.17×	1.16	33.37	16.18	0.70	0.39
TAP ($N = 5$)	34.21	3100.76	4.17 ×	1.19	33.55	18.72	0.78	0.27
Qwen-Image-Lightning (8 steps)	7.46	560.96	-	1.30	33.06	-	-	-
TaylorSeer ($N = 5, O = 2$)	4.68	<u>280.54</u>	2.00×	<u>1.21</u>	<u>33.03</u>	<u>17.59</u>	<u>0.70</u>	<u>0.27</u>
TAP ($N = 5$)	<u>4.80</u>	280.50	2.00 ×	1.27	33.50	21.11	0.77	0.23

they are more robust to abrupt, noncontinuous token dynamics and therefore complement higher-order predictors, yielding larger gains than using only high-order variants. Second, shifting the prediction window to the left (earlier expansion points) gives notable improvements because it avoids extrapolating beyond a token’s Taylor convergence radius; moving the window to the right (e.g., $[k, k+2]$) yields little benefit. These findings validate the assumptions and effectiveness of our method.

Comparison with Global Predictors. We constructed a pool of 30 candidate predictors by systematically varying Taylor expansion order and prediction horizon, and evaluated each predictor’s generative performance in isolation. The ImageReward of individual global predictors varied roughly between 0.86 and 0.92, while PSNR ranged approximately from 14.51 to 15.32, no single global predictor was best across all acceleration regimes. Instead, TAP’s probe-driven, per-token selection adaptively fuses these di-

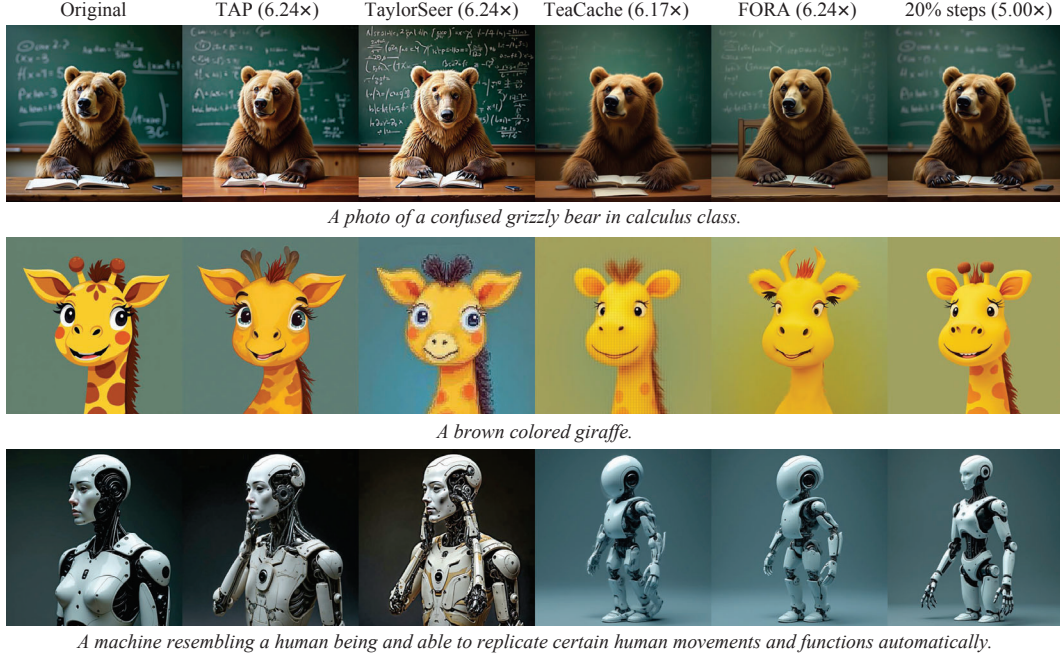


Figure 3. Visualization results. On FLUX.1-dev, TAP delivers higher speedup without quality loss.

Table 3. Quantitative comparison on VBench (HunyuanVideo).

Method	Latency (s) ↓	FLOPs (T)	VBench (%) ↑
50 steps	121.32	5481.50 _(1.00×)	66.61
50% steps	60.96	2740.75 _(2.00×)	<u>66.38</u>
20% steps	<u>24.80</u>	<u>1096.30</u> _(5.00×)	64.16
FORA ($N = 5$)	31.68	1101.72 _(4.98×)	63.87
TeaCache ($\ell = 0.4$)	32.33	1205.42 _(4.55×)	<u>65.13</u>
TeaCache ($\ell = 0.5$)	25.53	991.96 _(5.53×)	64.28
Taylor ($N = 6, O = 2$)	33.67	1101.72 _(4.98×)	64.89
TAP ($N = 6$)	31.91	<u>1101.47</u> _(4.98×)	65.46

verse predictors and consistently outperforms any individual predictor, showing that the improvements arise from intelligently combining complementary predictors rather than from a single optimal sampler.

4.4. Qualitative Analysis

We present representative visualizations in Figure 3 and Figure 4. Across both image and video examples, cache-only and global-forecast baselines exhibit noticeable degradation at high acceleration ratios, manifesting as blurred details, distorted object geometry, and misalignment with text conditions. In contrast, TAP effectively preserves fine-grained textures, structural integrity, and visual consistency. This advantage comes from TAP’s token-adaptive assignment, which selects the best predictor for each token at every timestep and thus preserves perceptual fidelity even at high acceleration ratios. These qualitative observations match the quantitative gains reported above.

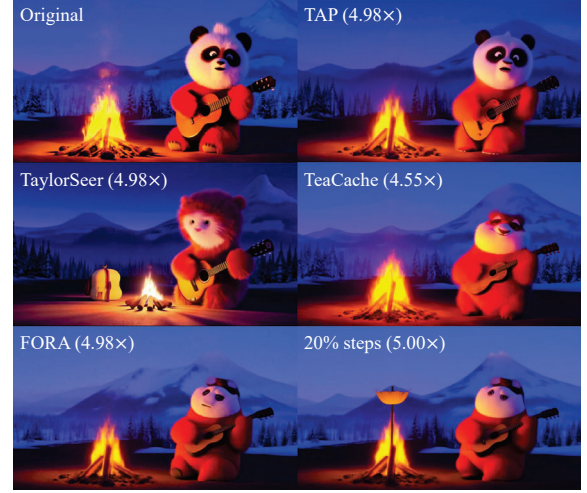


Figure 4. Visualization of video generation. “A happy fuzzy **panda** playing guitar nearby a campfire, snow mountain in the background”.

5. Conclusion

We present TAP, a probe-driven, token-adaptive diffusion-acceleration framework that makes per-token predictions based on a lightweight proxy. TAP is highly efficient, fully parallelizable, and compatible with a wide range of predictor designs. It achieves substantial speedups in diffusion sampling with minimal memory and compute overhead while preserving perceptual quality. Experiments demonstrate consistent gains across both image and video models.

References

- [1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 1
- [2] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4599–4603, 2023. 2
- [3] Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Tong Wu, Dahua Lin, and Jiaqi Wang. Dicache: Let diffusion model determine its own cache. *arXiv preprint arXiv:2508.17356*, 2025. 3
- [4] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. δ -dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024. 1, 2
- [5] Zeyi Cheng. Paraattention: Context parallel attention that accelerates dit model inference with dynamic caching. <https://github.com/chengzeyi/ParaAttention>, 2025. Accessed: YYYY-MM-DD. 5
- [6] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *arXiv preprint arXiv:2305.10924*, 2023. 2
- [7] 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. 6
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. arXiv:2006.11239 [cs]. 1
- [9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2
- [10] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024. 6
- [11] 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*, pages 1383–1392, 2024. 2
- [12] 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. 2
- [13] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 6
- [14] Black Forest Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024. 6
- [15] Eran Levin and Ohad Fried. Differential diffusion: Giving each pixel its strength. In *Computer Graphics Forum*, page e70040. Wiley Online Library, 2025. 5
- [16] 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)*, pages 17489–17499, 2023. 2
- [17] 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. 2
- [18] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It’s time to cache for video diffusion model, 2024. 2, 3, 5, 6
- [19] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From Reusing to Forecasting: Accelerating Diffusion Models with TaylorSeers. 1, 2, 6
- [20] Jiacheng Liu, Peiliang Cai, Qiming Zhou, Yuqi Lin, Deyang Kong, Benhao Huang, Yupei Pan, Haowen Xu, Chang Zou, Junshu Tang, et al. Freqca: Accelerating diffusion models via frequency-aware caching. *arXiv preprint arXiv:2510.08669*, 2025. 2, 4
- [21] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers, 2025. 4, 5
- [22] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. Specca: Accelerating diffusion transformers with speculative feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10024–10033, 2025. 2, 6
- [23] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 2
- [24] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 2
- [25] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15762–15772, 2024. 2
- [26] Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022. 2
- [27] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2023. arXiv:2212.09748 [cs]. 1
- [28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2022. arXiv:2112.10752 [cs]. 1

- [29] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 8(4):e1249, 2018. 1
- [30] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24219–24238, 2022. 6
- [31] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 2
- [32] 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. 1, 6
- [33] 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*, pages 1972–1981, 2023. 2
- [34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 2
- [35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 1, 2
- [36] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 6
- [37] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. 6
- [38] Jiazhen Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagere-ward: Learning and evaluating human preferences for text-to-image generation, 2023. 6
- [39] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. In *NeurIPS*, 2024. 2
- [40] Zhihang Yuan, Hanling Zhang, Lu Pu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. DiTFastattn: Attention compression for diffusion transformer models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2
- [41] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric, 2018. 6
- [42] 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. 2
- [43] Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers, 2025. 2
- [44] 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. 2
- [45] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024. 1, 2, 4, 5