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

Supplementary Material

1. Implementation Details

Model Specification The images generated by FLUX.1-dev, FLUX.1-schnell, and Qwen-Image-Lightning were produced at a resolution of 1024×1024, while outputs from Qwen-Image were produced at 1328×1328. Video frames from HunyuanVideo were obtained at 480×832. All reported measurements (performance, FLOPs, latency, and memory usage) are normalized to per-sample values at a batch size of 1. All metrics were collected on identical hardware across the evaluation dataset (e.g., 200 trials on DrawBench), with 3 additional warm-up inference runs performed before measurement.

FLOPs and HBM We measure FLOPs using the open-source tool `calflops`. For a Transformer with L layers, sequence length N_x and hidden dimension D, a compact estimate is FLOPs $\approx L\big(24N_xD^2 + 4N_x^2D\big)$, where the $24N_xD^2$ term approximates linear projections and FFN matmuls and the $4N_x^2D$ term captures attention matmuls. During reuse steps, only the modulated input layer (Eq. (5)) is recomputed, which reduces the overall FLOPs. Peak HBM grows roughly linearly with batch B and tokens N_x, a simple upper bound is $\mathrm{HBM}_{\mathrm{peak}}(B, N_x) \approx P \cdot b + (|C| + \alpha)\, B\, N_x\, D \cdot b$, where P is the parameter count, b is bytes per element (e.g. $b = 2$ for fp16), $|C|$ is the number of cached feature tensors (shallow probe and residual features), and α models transient activations. Prior caching methods require storing per-layer features, whereas TAP only caches the probe and residual features, thereby substantially reducing memory requirements for caching.

Baseline Specification We implement FORA, TeaCache, TaylorSeer, and SpeCa using their official codebases. In FORA, TaylorSeer, and TAP, N denotes the cache-window length: we compute on the first step and skip the remaining $N - 1$ steps within each window of N steps. In TaylorSeer, O denotes the maximum prediction order. In TeaCache, l is the threshold used to decide whether to compute or reuse cached features at the beginning of each step. In SpeCa, N and M denote the minimum and maximum caching steps, respectively, for each sample. Following prior works [5, 6], we retain the features computed in the previous three timesteps during sampling (two timesteps for FLUX.1-schnell), since preserving early computations is important for generation quality. The same protocol is applied to the TaylorSeer and SpeCa baselines. Besides, we use the official flow-matching sampler for step-reduced baselines (same setup across comparisons).

2. Differences from Existing Work

Our approach differs from prior training-free acceleration methods in several important ways:

Comparison to Direct-Reuse Methods Unlike direct feature-reuse methods [7, 8, 10], TAP leverages predictor-based forecasting to aggregate information from multiple past fully-computed steps and predict future features. Note that direct reuse is a special case of our predictor family (equivalent to a zero-order predictor), so TAP can exploit both direct-reuse and prediction-based strategies jointly.

Comparison to Prediction-based Methods Existing predictor-based accelerators such as TaylorSeer [5] and AB-Cache [9] typically rely on a single, fixed predictor. By contrast, TAP dynamically and organically combines multiple predictor types per token and per timestep, which improves generation quality. Methods that split features into frequency bands and apply fixed predictors per band (e.g., FreqCa [4]) are still limited by their use of fixed predictors. TAP's more diverse candidate set captures a wider range of temporal behaviors.

Comparison to Sample-Adaptive Methods Sample-adaptive methods like SpeCa [6] and TeaCache [3] allocate computation on a per-sample basis by deciding whether to compute or skip each step. While sample-adaptive strategies are powerful, they often break efficient batch-parallel inference due to varying per-sample workloads and typically require hand-tuned thresholds that must be re-tuned across models (e.g., TeaCache's polynomial coefficients). TAP is threshold-free and supports batch-parallel inference without manual threshold tuning. Besides, it can incorporate sample-adaptive ideas. For example, falling back to full computation when proxy losses are large.

Comparison to TaylorSeer TaylorSeer is an influential prediction-based method. We extend its insights in two ways. First, we observe that prediction behavior varies substantially across predictor order and distance, so we construct a Taylor predictor family to better capture per-token

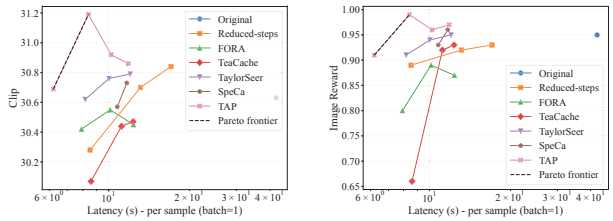


Figure 5. Pareto plots on FLUX.1-dev evaluated on DrawBench.

temporal heterogeneity. Second, TAP can benefit from predictors beyond Taylor expansions (see Table 4), demonstrating broader compatibility and improved robustness.

Summary In short, TAP unifies the strengths of existing approaches while avoiding their main limitations (fixed predictors, hand-tuned thresholds, or loss of batch parallelism). Empirically, TAP results in substantial improvements in generation quality and efficiency compared to prior methods.

3. More Quantitative Experiments

Pareto Plots To better illustrate the speed–quality tradeoff, we provide Pareto plots. As shown in Figure 5, we present latency vs. CLIP and latency vs. ImageReward curves on FLUX.1-dev to characterize the speed–quality frontier.

Summary of Hyperparameter Sensitivity We provide a comprehensive set of ablation experiments to justify our default hyperparameter choices. In particular, Figure 2 presents systematic sweeps over predictor order M and prediction distance λ, demonstrating the typical diminishing returns as order or prediction distance grows and identifying a stable operating region that balances accuracy and cost. We also evaluate the effect of selection granularity δ, as noted in the manuscript, "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." Finally, the cache window / acceleration control parameter N is swept in Tables 1–3, which report results for multiple values of N and thereby quantify the expected fidelity–speed tradeoff. Taken together, these experiments show that (i) moderate predictor orders and short-to-medium prediction distances yield the best tradeoff in practice, (ii) a coarse granularity $\delta = 1$ is a robust and simple default because finer granularity brings only marginal improvements, and (iii) N cleanly controls the acceleration ratio with predictable effects on quality as documented in Tables 1–3.

Alternative Predictors We introduce the Taylor predictor family to capture diverse temporal dynamics, and emphasize that TAP is compatible with other predictor families [4, 11]. To demonstrate this compatibility, we additionally evaluate a nonlinear predictor built from probabilists' Hermite polynomials [2, 4] with second-order prediction ($m = 2$). For a normalized time $t \in [-1, 1]$, the predicted feature $\widehat{\mathbf{r}}_t$ for token i is modeled as

$$\widehat{\mathbf{r}}_t^{(i)} = \sum_{k=0}^{m} c_{i,k}\, \mathrm{He}_k(t),$$

where $\{\mathrm{He}_k\}_{k=0}^{m}$ are Hermite polynomials and the coefficients $c_{i,k}$ are estimated by ordinary least-squares using the K most recent cached steps. We form a design matrix from the recent timestamps and solve for $c_{i,\cdot}$ that minimize the squared prediction error on those cached features.

The results in Table 4 show that combining TAP with the Hermite-based predictor yields additional gains beyond TAP alone, confirming that our framework is compatible with different feature-prediction methods and can benefit from richer predictor families.

Table 4. Comparison of alternative predictors. Experiments conducted on the FLUX.1-dev baseline; TAP with $N = 5$.

Method	ImageReward↑	PSNR↑
Baseline	0.95	16.69
TAP	<u>0.97</u>	<u>17.43</u>
TAP (w/ Hermite)	**0.98**	**17.52**

Memory Usage Analysis We report peak GPU memory usage for each backbone in Table 5. TAP introduces only 0.29 GB of additional memory (approximately 0.6% of the original model), whereas TaylorSeer requires 11.91 GB of extra memory (about 26.27%) on HunyuanVideo model. This demonstrates that TAP imposes negligible memory overhead while still providing substantial acceleration.

Table 5. Peak GPU memory usage (GB) during sampling.

Backbone	Original	TaylorSeer	TAP
FLUX.1-dev	37.72	45.17	37.82
FLUX.1-schnell	37.66	44.80	37.80
Qwen-Image	65.21	96.03	65.72
Qwen-Image-Lightning	63.04	69.26	63.14
HunyuanVideo	45.33	57.24	45.62

Choice of Probe Our method uses the timestep-embedding–modulated input as the probe for computing proxy losses. For comparison, we also evaluate the raw model input. Results are reported in Table 6. We find that the modulated input yields larger gains than the raw

input. We attribute this to the stronger correlation between the modulated input and the model output dynamics (i.e., the modulated input better reflects how features evolve over timesteps), which is consistent with the analysis in Figure 7.

Table 6. Comparison of probes. Experiments conducted on the FLUX.1-dev baseline; TAP with $N = 8$.

Method	ImageReward↑	PSNR↑
Baseline	0.90	14.70
TAP (input probe)	0.97	15.98
TAP (modulated-input probe)	**0.99**	**16.11**

Comparison of Proxy Losses We compare the cosine distance against the ℓ_1 loss as a proxy loss to quantify prediction errors. Results are presented in Table 7. Both losses deliver clear improvements over existing baselines, with the cosine proxy showing a stronger advantage compared to the ℓ_1 in our experiments.

Table 7. Comparison of proxy losses. Experiments conducted on the FLUX.1-dev baseline; TAP with $N = 8$.

Method	ImageReward↑	PSNR↑
Baseline	0.90	14.70
TAP (ℓ_1 loss)	0.97	16.05
TAP (cosine loss)	**0.99**	**16.11**

More Comparison with Global Predictors In the main manuscript we compare a single global predictor with our token-adaptive predictor. Then we further illustrate those results in Figure 6. As shown, TAP clearly outperforms a single global predictor under the same sampling budget and acceleration settings, demonstrating the benefit of per-token adaptation.

Human Evaluation We further include FID, a VQA score, and a human preference study for comprehensive evaluation, with all results reported in Table 8.

Table 8. With FLUX.1-dev on DrawBench, FID is computed between images from accelerated and original (unaccelerated) models. Preference is the fraction of sample-level judgments favoring the method, averaged over 10 human raters.

Method	FID ↓	VQAScore (%) ↑	Preference (%) ↑
TaylorSeer (N=8)	102.25	77.43	16.5
TAP (N=8)	**82.98**	**77.90**	**35.5**

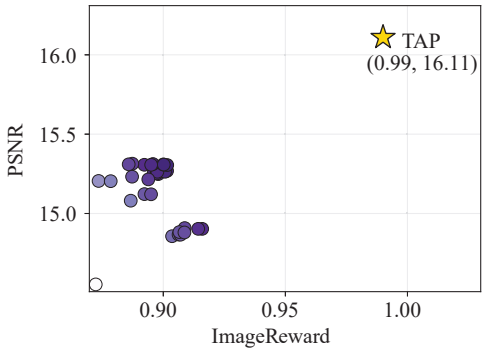


Figure 6. Comparison to global predictors. Experiments are conducted on FLUX.1-dev with 50 sampling steps and a fixed acceleration of $N = 8$. TAP achieves better performance than a single global predictor.

4. More Qualitative Experiments

Input-Output Analysis Our method estimates predictor performance from the prediction error of the model input, based on the strong relationship between a model's input and its output as shown in prior work [1, 3]. We visualize the differences of the input, the modulated input, and the model output between consecutive timesteps in Figure 7. The results show that both the raw input and the modulated input exhibit a strong correlation with the model output. In particular, the modulated input shows a stronger correlation, thus we choose it as our probe.

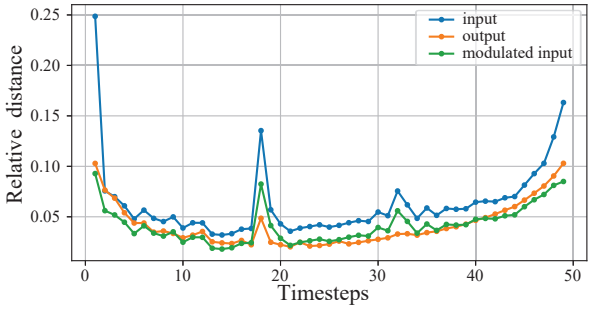


Figure 7. Correlation between probe proxy loss and downstream feature error. Strong positive correlation supports the probe-then-select design.

Token Heterogeneities Analysis We additionally analyze the relative error of token features over time. As shown in Figure 9, different tokens can exhibit markedly different temporal dynamics: for example, low-texture regions tend to vary smoothly across timesteps, while high-texture or content-rich regions may undergo abrupt changes at mid timesteps during generation. These observations motivate adaptive, per-token assignment of predictors to better match

each token's temporal behavior.

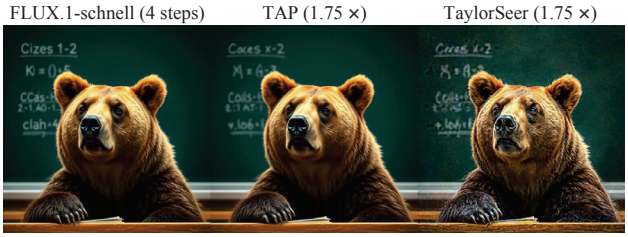


Figure 8. Visualization results on FLUX.1-schnell with 4 steps.

Token Prediction Patterns To further analyze the predictor assignments produced by TAP, we visualize TAP's averaged per-token predictions in Figure 10. We observe a clear temporal pattern. In the early stage (steps 0–18), the model predominantly selects lower-order predictors. In the mid-to-late stage (steps 18–50), the model shifts toward higher-order predictors and the variance of selected orders increases substantially, indicating a divergence in temporal-evolution patterns across tokens and motivating per-token assignment of different orders. A similar trend appears for prediction distance: its variance grows in the late stage (notably steps 42–50). We also find that the model prefers smaller distances during the early stage (0–18 steps) to remain closer to the convergence point. This behavior is expected because early-generation content is more uncertain, features evolve rapidly and non-smoothly, and Taylor-based forecasts computed from early steps suffer larger truncation errors and have a smaller radius of convergence. Together, these observations indicate that TAP dynamically adapts both predictor order and prediction horizon over time to better match changing generation dynamics.

Additional Qualitative Analysis We provide additional visualizations for the distilled variants in Figure 8 to enable direct visual comparison.Besides, further visualizations are presented in Figures 11–13. Compared with baseline methods, our approach produces results that are more faithful to the original model and better aligned with the text instructions. The baselines compared frequently exhibit visible distortions, loss of fine detail, or semantic misalignment.

References

[1] 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

[2] Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, and Linfeng Zhang. Hicache: Training-free acceleration of diffusion models via hermite polynomial-based feature caching, 2025. 2

[3] 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. 1, 3

[4] Jiacheng Liu, Peiliang Cai, Qinming 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. 1, 2

[5] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers, 2025. 1

[6] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. Speca: Accelerating diffusion transformers with speculative feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10024–10033, 2025. 1

[7] 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. 1

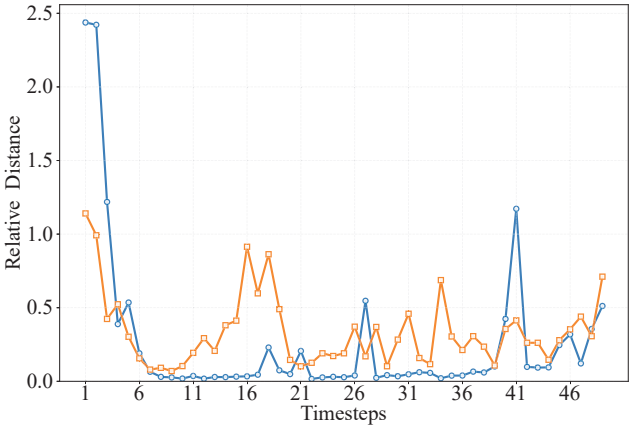
[8] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024. 1

[9] Zichao Yu, Zhen Zou, Guojiang Shao, Chenwei Zhang, Shengze Xu, Jie Huang, Feng Zhao, Xiaodong Cun, and Wenyi Zhang. Ab-cache: Training-free acceleration of diffusion models via adams-bashforth cached feature reuse. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10408–10417, 2025. 1

[10] 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. 1

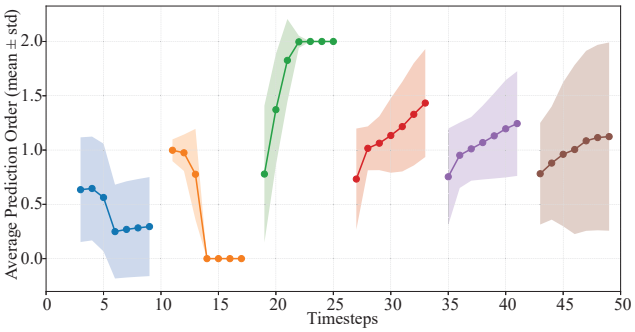
[11] Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers, 2025. 2

(a) Generation example.

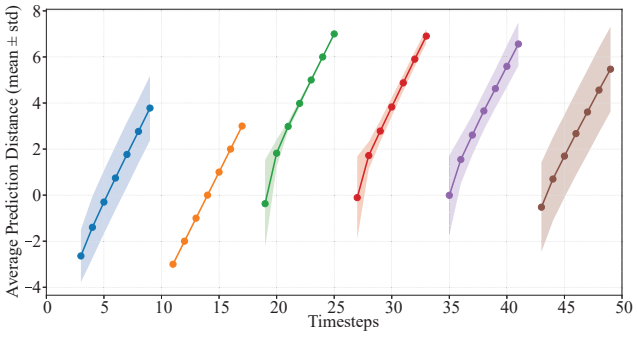


(b) Token distance across adjacent timesteps.

Figure 9. Analysis of token evolution patterns. Different tokens follow different temporal patterns: low-texture regions (blue token) change smoothly across timesteps, whereas content-rich regions (orange token) exhibit discontinuous and larger variations around mid timesteps. Results were obtained on FLUX.1-dev with 50 sampling steps.



(a) Average selected prediction order over timesteps.



(b) Distribution (mean and variance) of the selected prediction horizon over timesteps.

Figure 10. Prediction patterns generated by TAP. Experiments were conducted on FLUX.1-dev with 50 sampling steps and an acceleration factor of $N = 8$.

Figure 11. Visualization results of FLUX.1-dev.

Figure 12. Visualization results of Qwen-Image.

A koala bear playing piano in the forest. *a cow on the right of an elephant, front view*

Figure 13. Visualization results of HunyuanVideo.