ResCa: Residual Caching for Diffusion Transformers Acceleration

Supplementary Material

We organize our supplementary material as follows:

A. Implementation Details

A.1. Details of Preliminary Experiments

In Sec. 3, we further investigate our "proxy denoising" idea by addressing two key questions: where to find the similar residuals and how to use the similar residuals. Below, we detail the settings of the preliminary experiments.

Feature Collection. We collect features from the FLUX.1-dev model based on 10,000 prompts. For each timestep t and each block l, we extract the feature $x^l_{k,t}$ for every sample k. We adopt the model's default setting of 50 sampling steps with 1024×1024 resolution. The 10,000 prompts consist of 5,000 prompts from COCO 2014 validation set and 5,000 prompts from COCO 2017 validation set.

Trajectory Approximation. We adopt the trajectory approximation method [3] to approximate the high-dimensional sampling trajectory within a 3-D subspace spanned by a primary direction $\mathbf{u}$ and two principal directions $\mathbf{v}_1$ and $\mathbf{v}_2$. For notational simplicity, we omit the sample index k and the layer index l in the following and use boldface to denote vectors. Therefore, we write $\mathbf{x}_t$ instead of $x^l_{k,t}$. Let $\{\mathbf{x}_{t_i}\}_{i=0}^T$ represent a trajectory in $\mathbb{R}^d$ with endpoints $\mathbf{x}_{t_0}$ and $\mathbf{x}_{t_T}$. We first define the main direction as

$$\mathbf{u} = \frac{\mathbf{x}_{t_T} - \mathbf{x}_{t_0}}{\|\mathbf{x}_{t_T} - \mathbf{x}_{t_0}\|_2}. \tag{S1}$$

Next, we center the trajectory as $\tilde{\mathbf{x}}_{t_i} = \mathbf{x}_{t_i} - \mathbf{x}_{t_0}$. Then, we project $\tilde{\mathbf{x}}_{t_i}$ onto the subspace orthogonal to $\mathbf{u}$, perform PCA on this projection, and extract the first two principal components $\mathbf{v}_1$ and $\mathbf{v}_2$. Each sample is then approximated in the 3-D subspace spanned by $\mathbf{u}$, $\mathbf{v}_1$, and $\mathbf{v}_2$ as

$$\tilde{\mathbf{x}}_{t_i} \approx (\tilde{\mathbf{x}}_{t_i} \cdot \mathbf{u})\,\mathbf{u} + (\tilde{\mathbf{x}}_{t_i} \cdot \mathbf{v}_1)\,\mathbf{v}_1 + (\tilde{\mathbf{x}}_{t_i} \cdot \mathbf{v}_2)\,\mathbf{v}_2, \tag{S2}$$

where $(\tilde{\mathbf{x}}_{t_i} \cdot \mathbf{u})$, $(\tilde{\mathbf{x}}_{t_i} \cdot \mathbf{v}_1)$, and $(\tilde{\mathbf{x}}_{t_i} \cdot \mathbf{v}_2)$ denote the projections of $\tilde{\mathbf{x}}_{t_i}$ onto the corresponding directions. Approximating the trajectory by its projection onto the 3-D subspace spanned by $\mathbf{u}$, $\mathbf{v}_1$, and $\mathbf{v}_2$ provides a faithful low-dimensional representation of the original high-dimensional path, since these three directions together explain approximately 85% of the total variance [3] and preserve the essential geometric structure of the sampling trajectories.

Experiment Details for the First Question. As shown in the upper part of Fig. 2, we conduct three analyses: (a) illustration of feature-based clusters, (b) illustration of trajectory-based clusters, and (c) a comparison of clustering methods. For (a) and (b), we use the trajectory approximation introduced above to visualize token-wise denoising paths. We plot the trajectories of 12 tokens, color-coded by cluster (purple, blue, and red). Among the 50 sampling steps, darker points indicate the first 46 timesteps that are actually sampled, while lighter points denote the remaining 4 unsampled steps, highlighting the differences along the denoising trajectory. For (c), for each cluster C, we define the first-order residual as $\Delta^{(1)}x^l_{k,t} = x^l_{k,t} - x^l_{k,t+1}$ and compute the pairwise ℓ_2 distance within the cluster as

$$d^{(m)}_{i,j} = \left\| \Delta^{(m)} x^l_{k,t,i} - \Delta^{(m)} x^l_{k,t,j} \right\|_2, \quad i,j \in C, ; i \neq j. \tag{S3}$$

We aggregate $\{d^{(1)}_{i,j}\}$ over all timesteps, layers, and clusters, normalize both the distances and their counts, and obtain the corresponding distance–frequency statistics. The largest 1% of distances are clipped for visualization.

Experiment Details for the Second Question. As shown in the lower part of Fig. 2, we conduct two analyses: (d) intra-cluster ℓ_2 distance of multi-order residuals and (e) intra-cluster residual relationships across timesteps. For (d), we define the m-th residuals as

$$\Delta^{(m)}x^l_{k,t} = \begin{cases} x^l_{k,t}, & m = 0, \\ x^l_{k,t} - x^l_{k,t+1}, & m = 1, \\ \Delta^{(m-1)}x^l_{k,t} - \Delta^{(m-1)}x^l_{k,t+1}, & m \geq 2. \end{cases} \tag{S4}$$

and then compute the pairwise ℓ_2 distance of m-th order residuals within each cluster following the definition in Eq. S3. We then aggregate the resulting distances $\{d^{(m)}_{i,j}\}$ over all timesteps, layers, and clusters, normalize the distances, and obtain the order–distance box plot. From top to bottom, the upper whisker, box top, box bottom, and lower

whisker correspond to the 98%, 90%, 10%, and 2% quantiles, respectively. For (e), we visualize the relationship between the multi-order residual distances at timestep t and those at the adjacent timestep $t+1$. The largest 2% of distances are clipped for visualization.

A.2. Model Configuration in Experiments

Following the settings of TaylorSeer [22], we evaluate the effectiveness of our method on three generative tasks: class-conditional image generation with DiT-XL/2 [30], text-to-image generation with FLUX.1-dev [17], and text-to-video generation with HunyuanVideo [16]. For comparison, we include eight representative baselines, including Δ-DiT [5], Chipmunk [36], FORA [34], `ToCa` [51], `DuCa` [52], TeaCache [20], TaylorSeer [22], and ClusCa [49]. The corresponding model configurations are summarized below.

FLUX.1-dev. We adopt the default Rectified Flow sampler with 50 denoising steps and uniformly sample 200 prompts from DrawBench [31], following the standard comparative protocol. Image quality is assessed using Image Reward [44], while text–image alignment is measured with the CLIP score [11]. To measure efficiency, we report both latency and FLOPs, where latency is measured on NVIDIA A800 80GB GPUs. As baselines, we consider configurations with 17 and 11 sampling steps and tune the hyperparameters of competing methods to achieve comparable acceleration ratios. For qualitative analysis, we further include prompts from PartiPrompts and the MS-COCO 2017 validation set to increase prompt diversity.

HunyuanVideo. We build upon the native HunyuanLarge architecture and adopt a standardized 50-step inference protocol, keeping all official hyperparameters unchanged to ensure sampling consistency across compared models. Video quality is evaluated using the VBench [13] framework, where we generate five videos per prompt over 946 distinct prompts, each with a different random seed. Efficiency is measured by both latency and FLOPs, with latency recorded on NVIDIA H20 96GB GPUs. As baselines, we employ sampling schedules with 11 steps, and for all competing methods we tune the corresponding hyperparameters to obtain similar acceleration ratios, thereby enabling fair group-wise comparisons.

DiT-XL/2. For class-conditional image generation with DiT-XL/2, we use the default DDIM sampler with 50 denoising steps, following the established comparative setting. We uniformly sample 50,000 images from all 1,000 ImageNet classes at a resolution of 256×256. Image quality is primarily assessed using FID-50k [12], and we additionally report sFID as a complementary metric. Efficiency is evaluated by reporting both latency and FLOPs, where latency is measured on NVIDIA A800 80GB GPUs. As DDIM baselines with a small number of sampling steps, we adopt sampling schedules with 25, 20, 15, and 12 steps, and adjust the hyperparameters of various caching methods, which facilitates fair comparisons within each group.

A.3. Details of PDDS with Implicit ODE Solvers

The implicit ODE Solvers. The reverse-time dynamics of a diffusion model can be written in the continuous-time limit as an ordinary differential equation:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f_\theta\big(x(t), t\big), \tag{S5}$$

where $x(t)$ is the latent state at time t, and f_θ is a drift field induced by the score network. For a unit reverse-time step from t to $t-1$, we first write a Taylor expansion of $x(t)$ around $t-1$ with step size $\Delta t = 1$:

$$x(t) = x(t-1) + \sum_{m=1}^{\infty} \frac{1}{m!} \frac{\mathrm{d}^m x(t-1)}{\mathrm{d}t^m}. \tag{S6}$$

Rearranging terms gives an exact continuous-time implicit Taylor representation:

$$x(t-1) = x(t) - \sum_{m=1}^{\infty} \frac{1}{m!} \frac{\mathrm{d}^m x(t-1)}{\mathrm{d}t^m}. \tag{S7}$$

If we approximate the m-th time derivative at $t-1$ by a discrete operator $D^{(m)}(\cdot)$ and truncate the series at order M, we obtain the classical implicit Taylor scheme

$$x(t-1) = x(t) - \sum_{m=1}^{M} \frac{1}{m!} D^{(m)}\big(x(t-1)\big), \tag{S8}$$

where $D^{(m)}(x)$ is a discrete approximation of the m-th time derivative evaluated at time $t-1$.

Combination with implicit ODE Solvers. In our method, the driven token d_t follows exactly the same structure, with the discrete derivatives given by the proxy-driven multi-order residuals $\hat{\mathcal{F}}^{(m)}(d_{t-1})$:

$$d_{t-1} = d_t - \sum_{m=1}^{M} \frac{1}{m!} \hat{\mathcal{F}}^{(m)}(d_{t-1}), \tag{S9}$$

which is an instance of the implicit Taylor scheme in Eq. (S8). For driven tokens, we deliberately avoid extra evaluations of the diffusion network and thus do not have direct access to the drift f_θ. Instead, the multi-order residuals $\hat{\mathcal{F}}^{(m)}(d_{t-1})$ serve as discrete approximations to the time derivatives at $t-1$, and they already depend on the unknown state d_{t-1} through the proxy correction at time $t-1$. This structural dependence makes an implicit formulation natural: the right-hand side in Eq. (S9) is a function of d_{t-1}, and solving Eq. (S9) is equivalent to taking one reverse-time integration step of a surrogate ODE whose drift is represented

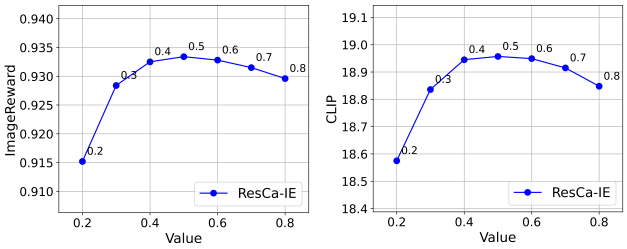


Figure S1. **Ablation** on the temporal smoothing factor.

Model	Clustering	Latency(s)	FLOPS(T)	Quality
FLUX.1	feature-based	6.78	675.0	0.9536
	trajectory-based	6.79	675.1	0.9737
HunyuanVideo	feature-based	83.38	5382.9	78.52
	trajectory-based	83.42	5383.2	79.98

Table S1. **Ablation** on Clustering Methods. The quality metric is DrawBench for FLUX.1 and VBench for HunyuanVideo.

by the proxy-driven residuals. More formally, we can view the driven token as following a surrogate ODE:

$$\frac{\mathrm{d}d(t)}{\mathrm{d}t} = \hat{f}\big(d(t), t\big) \triangleq -\sum_{m=1}^{M} \frac{1}{m!}\hat{\mathcal{F}}^{(m)}\big(d(t)\big), \qquad \text{(S10)}$$

where $\hat{f}$ is an approximation to the true drift f_θ constructed from proxy-driven residuals. Applying the implicit Taylor step with step size 1 to Eq. (S10) leads exactly to Eq. (S9), so our update is a consistent discretization of this surrogate dynamics. Further analysis is provided in Sec. B.8.

B. Additional Experiments and Analyses

B.1. Ablation on Temporal Smoothing Factor

In Step 2 of Temporal-Enhanced Trajectory Clustering, we accumulate the similarity across timesteps. To balance the contribution of the similarity at the current timestep and the accumulated historical similarity, we introduce a temporal smoothing factor $\alpha_{\mathcal{S}}$. A larger $\alpha_{\mathcal{S}}$ assigns more weight to the current timestep, whereas a smaller value places greater emphasis on the similarity of historical trajectories. Under the FLUX.1-dev configuration, where ResCa-IE is integrated with $\mathcal{N} = 8$, $O = 1$, and $K = 16$, we conduct an ablation study on $\alpha_{\mathcal{S}}$, as shown in Fig. S1. The results indicate that setting $\alpha_{\mathcal{S}} = 0.5$ yields the best performance for both ImageReward and CLIP. Increasing $\alpha_{\mathcal{S}}$ from 0.5 to 0.8 leads to a slight decrease in performance, which we attribute to insufficient use of historical trajectory information. In contrast, decreasing $\alpha_{\mathcal{S}}$ from 0.5 to 0.2 causes a marked degradation, since the direction of the current trajectory has already drifted from that of the historical trajectories, and over-reliance on outdated history substantially increases the clustering error.

Module	Step	Computation	Memory
TETC	Step 1	$\mathcal{O}(N^2D)$	$\mathcal{O}(ND+N^2)$
	Step 2	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
	Step 3	$\mathcal{O}(IN^2)$	$\mathcal{O}(N^2)$
PDDS	Step 2	$\mathcal{O}\big((N-K)MD\big)$	$\mathcal{O}(MND)$
	Step 3	$\mathcal{O}\big((N-K)MD\big)$	$\mathcal{O}(MND)$

Table S2. **Computation and memory complexity** of each step.

B.2. Ablation on Clustering Methods

In Sec. 3, we show that historical denoising trajectories can better group tokens with similar residuals than features at a single timestep. Here, we further provide a quantitative comparison in Table. S1. Specifically, we replace the proposed trajectory-based clustering in TETC with a feature-based variant that clusters tokens according to feature similarity at the current timestep, while keeping all other settings unchanged. We evaluate this comparison under the FLUX.1-dev setting with ResCa-IE integrated using $\mathcal{N} = 7$, $O = 1$, and $K = 16$, and under the HunyuanVideo setting with $\mathcal{N} = 6$, $O = 1$, and $K = 32$. As shown in Table S1, trajectory-based clustering consistently achieves better generation quality than feature-based clustering, with nearly identical latency and FLOPS on both FLUX.1 and HunyuanVideo. These results further support our motivation that historical multi-step dynamics provide a more reliable basis for identifying similar residuals.

B.3. Complexity and Latency Analysis

In ResCa, we introduce Temporal-Enhanced Trajectory Clustering (TETC) and Proxy-Driven Denoising Simulation (PDDS), corresponding to the dense and sparse computation stages, respectively. Although PDDS avoids the heavy computation of forwarding the $N - K$ driven tokens through the full blocks, these two modules also introduce additional overhead, including trajectory clustering, cached trajectory, residual propagation, and implicit ODE updates. In this section, we therefore provide a joint analysis of the computation complexity, memory cost, and latency of these components.

Complexity. For the feature at each timestep $X_t \in \mathbb{R}^{N\times D}$, where N denotes the number of tokens, D the feature dimension, K the number of clusters, I the number of K-medoids iterations, and M the maximum residual order, we summarize the per-step computation and memory complexity in Table S2. The heaviest component in TETC is Step 1, which computes pairwise similarity with cost $O(N^2D)$ and memory $\mathcal{O}(ND+N^2)$. However, compared with a standard MHSA layer that requires roughly $4N^2D + 2ND^2$ FLOPs, this overhead is still small and does not involve the expensive ND^2 term. Moreover, TETC is executed only once at the end of the dense computation stage in each group of $\mathcal{N}+1$ timesteps, so its amortized cost remains limited.

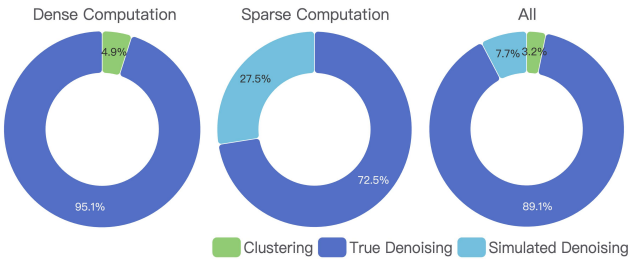


Figure S2. **Latency** of each operation. The additional clustering and simulated denoising introduce only small latency overhead.

For PDDS, the additional computation of residual estimation and implicit ODE update scales linearly as $O((N-K)MD)$, which is much smaller than forwarding all driven tokens through the transformer blocks. Its memory cost is $\mathcal{O}(MND)$, mainly due to caching multi-order residual trajectories for subsequent propagation. Therefore, PDDS is the main memory bottleneck in ResCa. Still, this memory overhead grows linearly with token count and residual order, and in practice remains manageable under the small M used in our implementation. Overall, the main speedup of ResCa comes from the sparse stage, where full denoising of the $N-K$ driven tokens is skipped, while the newly introduced operations only add moderate computation and memory overhead.

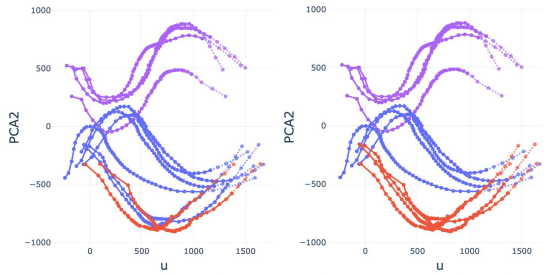
Latency. Beyond the theoretical complexity analysis, we further report the latency of each module. Specifically, we group TETC Steps 1–3 into the *clustering* operation, PDDS Steps 2–3 into *simulated denoising*, and refer to the remaining model computation as *true denoising*. In Fig. S2, we show the latency of each component over dense computation timesteps, sparse computation timesteps, and all timesteps. For this analysis, ResCa-IE is integrated with $\mathcal{N}=6$, $O=2$, and $K=16$. The results indicate that, during dense computation timesteps, *clustering* accounts for only about 4.9% of the total latency, since it is executed only once after the final block within each group of dense timesteps and thus incurs limited cost. During sparse computation timesteps, *true denoising* contributes roughly 72.5% of the latency, while *simulated denoising* occupies the remaining 27.5%. This shows that, even with a relatively small K, the latency spent on *simulated denoising* for the $(N-K)$ driven tokens remains considerably lower than that of *true denoising* for the K proxy tokens. When aggregating over all timesteps, the additional latency introduced by ResCa stays within 11%, whereas the time saved by reducing full diffusion computation is substantial (as shown in Table 1). Overall, the latency speedup reaches up to $3.6\times$, which supports the effectiveness of our design.

B.4. Comparison with Merging-based Methods

In the main manuscript, we primarily compare ResCa with caching-based approaches. In this section, we further include merging-based methods under the DiT-XL/2 setting. Because the acceleration of merging-based methods is constrained by the optimal merging ratio, they offer only limited acceleration. In particular, ToMeSD and AT-EDM achieve roughly $1.5\times$ speedup but suffer from noticeable degradation in generation quality. `SDTM` refines the merging strategy and partially leverages caching, increasing the speedup to about $1.6\times$ while alleviating the quality drop. In contrast, ResCa-IE attains a $1.8\times$ speedup with negligible loss in generation quality, which further highlights the advantage of our method.

Method	Latency(s) ↓	Speed ↑	FID ↓	sFID ↓
DDIM-50 steps	0.428	1.00×	2.32	4.32
ToMeSD	0.293	1.46×	2.74	4.94
AT-EDM	0.289	1.48×	2.64	4.82
`SDTM`	0.271	1.58×	2.53	4.72
ResCa-IE	0.234	**1.83**×	**2.37**	**4.63**

Table S3. **Additional comparison** with merging-based methods.



(a) Feature-based (b) Trajectory-based

Figure S3. **2D visualization** of denoising trajectories. Trajectory-based clustering yields higher intra-cluster similarity.

B.5. 2D Visualization of Denoising Trajectories

To make the trajectory trends in Fig. 2 (a) and (b) easier to perceive, we further provide a 2D visualization of the denoising trajectories in Fig. S3. Compared with the original 3D plots, the 2D projection offers a clearer view of the relative trajectory patterns among tokens. As shown in Fig. S3, feature-based clustering may group together tokens whose trajectories are not well aligned over time, leading to less consistent clusters. In contrast, trajectory-based clustering produces more compact and coherent trajectory groups, which better matches our motivation in Sec. 3 that historical denoising trajectories provide a more reliable basis for identifying similar residuals.

B.6. Visualization of Denoising Trajectory

To provide a more intuitive comparison between ResCa and representative methods, we randomly sample two token-wise denoising trajectories and visualize in Fig. S4, the corresponding "ground-truth" trajectories from the original FLUX model, together with those produced by a stan-

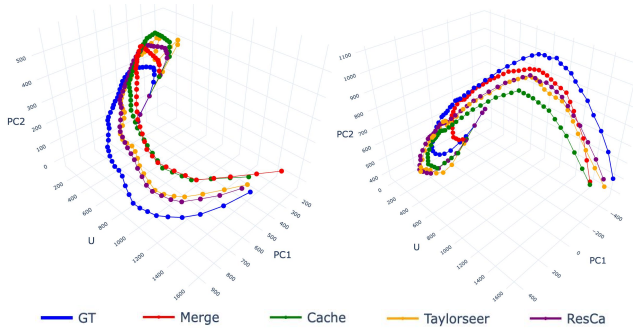


Figure S4. **Comparison** on the denoising trajectory. ResCa remains closer to the original denoising trajectory than others.

dard merging method, a standard caching method, the TaylorSeer, and our ResCa.

As the diffusion process progresses, the deviation of each approximate trajectory from the original denoising trajectory increases steadily, reflecting the accumulation of numerical errors that gradually drive the solution away from the target. Among all methods, the merging-based approach exhibits the largest final discrepancy from the ground-truth trajectory, and its trajectory and curvature, differs markedly from that of the original process. This observation is consistent with the finding in Table S3 that token merging typically struggles to deliver high acceleration without incurring substantial quality degradation.

Within caching-based approaches, the basic caching baseline still remains relatively far from the target trajectory. In contrast, the trajectory produced by our implicit-Taylor-based ResCa is clearly closer to the original trajectory than that obtained by the explicit-Taylor-based TaylorSeer. The key reason is that TaylorSeer relies solely on historical information for explicit extrapolation, whereas ResCa exploits proxy tokens to continuously refine the direction of the driven tokens' trajectories. These results show that ResCa more faithfully tracks the original denoising dynamics throughout sampling, thereby achieving a superior trade-off between generation quality and inference speed, as also evidenced in Table 1 of the main manuscript.

B.7. Human Preference Study

To complement the automatic metrics in the main paper, we further conduct a human preference study. Specifically, we recruit 20 volunteers with experience in visual content evaluation, including graduate students and researchers familiar with generative results, and evaluate 50 generated images and 50 generated videos. Participants are asked to compare different methods in terms of visual quality and prompt alignment. To reduce potential bias, the compared results are presented in random order with anonymous method names. We discard incomplete or invalid responses and report the final preference ratios based on the remaining valid votes. As shown in Fig. S5, ResCa achieves the highest human preference in both aspects. In particular, ResCa obtains 54.7% preference on visual quality and 42.1% on prompt alignment, consistently outperforming all compared methods. These results are consistent with the trends observed in the automatic metrics, and further suggest that the improvements of ResCa are also reflected in human perception.

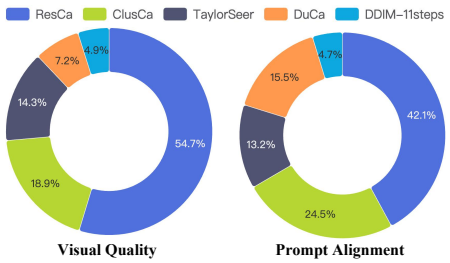


Figure S5. **Human Preference Study**. ResCa achieves the highest preference in both visual quality and prompt alignment.

B.8. Comparison of TaylorSeer and ResCa

Theoretical Comparison of *A*-Stability. Explicit Taylor predictors that only use history at t_n tend to amplify errors when the step size h is large, especially for stiff reverse-time dynamics. In contrast, our implicit Taylor update remains stable in such regimes and therefore controls the global error better in few-step diffusion sampling. The difference is most clearly seen on the standard linear test equation, which is the local linearization of general reverse-time dynamics:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \lambda y, \qquad \Re(\lambda) \le 0. \tag{S11}$$

(1) For explicit Taylor with $M = 1$, the history-only explicit Taylor step is

$$y_{n-1}^{\exp} = y_n^{\exp} + h\lambda y_n^{\exp} = (1 + h\lambda)\, y_n^{\exp}, \tag{S12}$$

the amplification factor is

$$G_{\exp}(z) = 1 + z, \quad z = h\lambda. \tag{S13}$$

Stability requires $|G_{\exp}(z)| \le 1$. For real $\lambda < 0$, we have $z = -\alpha h$ with $\alpha > 0$, so

$$G_{\exp}(z) = 1 - \alpha h. \tag{S14}$$

Whenever $\alpha h > 2$ we get $|1 - \alpha h| > 1$, and any small error is *magnified* at each step.

(2) For implicit Taylor with $M = 1$, our implicit Taylor update becomes backward Euler:

$$y_{n-1}^{\mathrm{imp}} = y_n^{\mathrm{imp}} + h\lambda y_{n-1}^{\mathrm{imp}}, \tag{S15}$$

so

$$(1 - h\lambda)\, y_{n-1}^{\mathrm{imp}} = y_n^{\mathrm{imp}}, \qquad y_{n-1}^{\mathrm{imp}} = \frac{1}{1 - h\lambda}\, y_n^{\mathrm{imp}}, \tag{S16}$$

Figure S6. **Qualitative comparison** between TaylorSeer and ResCa on FLUX.1-dev. As $\mathcal{N}$ increases, both methods exhibit degradation, but ResCa more faithfully preserves visual quality, whereas TaylorSeer shows stronger color distortion, blur, and semantic confusion.

the amplification factor is

$$G_{\text{imp}}(z) = \frac{1}{1-z}, \quad z = h\lambda. \tag{S17}$$

To check stability for all $\Re(\lambda) \le 0$, $\Re(z) \le 0$ implies

$$|1-z|^2 = (1-\Re(z))^2 + (\Im z)^2 \;\ge\; 1, \tag{S18}$$

hence $|1-z| \ge 1$ and

$$\left|G_{\text{imp}}(z)\right| = \frac{1}{|1-z|} \le 1. \tag{S19}$$

Thus backward Euler is A-stable: it never amplifies errors for any $h > 0$ and any λ with $\Re(\lambda) \le 0$.

For the same order M, explicit Taylor predictors and our implicit Taylor update have similar local accuracy, but: (i) the explicit, history-only scheme has amplification factor $G_{\text{exp}}(z) = 1 + z$ and becomes unstable when $h|\lambda|$ is large; (ii) our implicit Taylor update has $G_{\text{imp}}(z) = 1/(1-z)$ and is A-stable for $\Re(\lambda) \le 0$. In stiff reverse-time denoising with few large steps (the accelerated diffusion regime), this means the explicit predictor tends to accumulate and amplify proxy errors, while the implicit update keeps them bounded, leading to better global error control.

Qualitative Comparison with Large Caching Intervals. We present a comparison between the explicit-Taylor-based TaylorSeer and our implicit-Taylor-based ResCa in Fig. S6, showing generated results under different caching intervals where $\mathcal{N}$ ranges from 6 to 13. As $\mathcal{N}$ increases, both methods exhibit noticeable degradation, but ResCa consistently maintains higher visual fidelity, whereas TaylorSeer suffers from stronger color shifts, blurring, and semantic confusion. This behavior is consistent with our A-stability analysis: the explicit, history-only predictor tends to amplify errors at large step sizes, whereas the implicit update keeps them bounded. As a result, under large caching intervals, ResCa accumulates less error than TaylorSeer and delivers more stable generation quality.

B.9. Limitations and Future Work

Although ResCa achieves a strong trade-off between acceleration and generation quality, it still has several limitations: (1) *Model configuration.* ResCa introduces several hyperparameters, such as the caching interval $\mathcal{N}$, the number of clusters K, the residual order O, and the temporal smoothing factor $\alpha_{\mathcal{S}}$, which are empirically selected; in future work, we plan to investigate adaptive or data-driven strategies that automatically set these values. (2) *K-medoids efficiency.* The TETC module relies on K-medoids, an iterative clustering algorithm that can be relatively slow; a natural extension is to explore more efficient approximate or online clustering schemes to reduce the per-iteration cost. (3) *Implementation and memory overhead.* ResCa requires extra modules and explicit caching of intermediate states, which increases system complexity, GPU memory usage, and integration effort compared with standard sampling; in future work, we plan to develop lighter-weight, memory-aware variants and a streamlined implementation that integrates more naturally with existing diffusion inference pipelines.