TRAC: TRAJECTORY-AWARE REUSE AND ADAPTIVE CORRECTION FOR EFFICIENT AUTOREGRESSIVE VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive (AR) video generation enables scalable long-video synthesis by sequentially generating fixed-length chunks, but incurs substantial computational cost from repeated denoising and attention over growing historical contexts. Recent cache reuse methods exploit temporal redundancy to reduce this cost, yet are mainly designed for single-trajectory generation, making them less suitable for sequentially coupled AR trajectories where approximation errors can accumulate and propagate across chunks. To address this challenge, we propose **TRAC**, a **T**rajectory-aware **R**euse and **A**daptive **C**orrection framework for efficient and robust AR video generation. TRAC integrates three complementary components: (1) **Robust Cumulative Scheduling (RCS)**, which selects cache reuse schedules by accounting for cumulative rollout errors and their variation across chunks and prompts; (2) **Autoregressive Trajectory-aware Guidance Scheduling (ATGS)**, which coordinates classifier-free guidance refreshes along the global AR trajectory; and (3) **Spectral Structure Correction (SSC)**, which adaptively restores low-frequency structural information to mitigate long-term error accumulation. By combining these innovations, TRAC substantially improves the efficiency-quality trade-off of AR video generation. For example, on VBench with SkyReels-V2, it achieves a $6.03\times$ speedup over the vanilla model with only $\sim 0.6\%$ VBench score degradation, while consistently outperforming existing AR acceleration methods in both latency and generation quality.

1 INTRODUCTION

Autoregressive (AR) video generation has recently emerged as a promising paradigm for synthesizing long videos (Yin et al., 2025; Huang et al., 2026). Instead of denoising an entire video jointly, AR video models divide a long video into fixed-length chunks and generate them sequentially. For each chunk, the model performs a denoising process conditioned on the previously generated chunks, which are retained as historical context, making long-video generation substantially more scalable (Chen et al., 2025; Teng et al., 2025; Gao et al., 2024b).

However, this scalability comes at a considerable computational cost (Ma et al., 2026b; Ji et al., 2026; Lu et al., 2026). As generation proceeds, each new chunk must repeatedly attend to the growing historical context and undergo a heavy denoising trajectory. Moreover, classifier-free guidance (CFG) requires both conditional and unconditional predictions at each guided denoising step (Ho & Salimans, 2022), further increasing the inference cost. Consequently, long-video generation remains far from practical real-time deployment (Teng et al., 2025; Ma et al., 2026b).

To reduce this cost, a growing body of work accelerates video generation by selectively executing the full model at a subset of denoising steps and reusing cached information at the remaining steps (Guo et al., 2026; Ji et al., 2026; Shen et al., 2026; Lv et al., 2026; Yi et al., 2025; Lu et al., 2026; Ma et al., 2026b; Xu et al., 2026). Among them, training-free cache reuse-based acceleration methods are particularly attractive because they introduce no additional training or modification to the base generator (Ma et al., 2024b;a; Liu et al., 2025; Zhao et al., 2025; Kahatapitiya et al., 2025; Lyu et al., 2025; Liu et al., 2026; Zhou et al., 2025; Xi et al., 2025; Castillo et al., 2025; Lee et al., 2026; Gao et al., 2026). For example, MeanCache (Gao et al., 2026) approximates intermediate veloci-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ties using cached Jacobian-vector products (JVPs) and determines a cache schedule through offline calibration. CFG reuse methods (Castillo et al., 2025; Lyu et al., 2025; Lee et al., 2026) exploit the similarity between conditional and unconditional predictions by refreshing the guidance residual only at selected steps. These methods demonstrate that substantial redundancy exists throughout the generation process and can be effectively exploited for efficient inference.

Despite this progress, directly applying existing cache reuse methods to AR video generation remains challenging. Most existing methods are developed for bidirectional video models, where the video is generated through a single denoising trajectory and reuse errors are primarily confined to that trajectory (Ma et al., 2026b; Xu et al., 2026). In contrast, AR generation consists of sequentially connected chunk-level trajectories, where the approximation error in an earlier chunk can propagate through the autoregressive context and affect subsequent chunks. In this paper, considering this distinction, we identify three complementary challenges. **First, cache scheduling should account for cumulative and cross-sample errors rather than local similarity alone.** Existing schedulers such as MeanCache (Gao et al., 2026) mainly evaluate local reuse errors, which may overlook their accumulation along the generation path and their variation across chunks and prompts. We therefore introduce robust cumulative scheduling (RCS), which selects cache schedules by jointly considering cumulative rollout error and cross-chunk/prompt risk. **Second, CFG reuse should account for the sequential structure of the global AR trajectory.** Existing CFG reuse methods (Lyu et al., 2025; Lee et al., 2026) are mainly designed for bidirectional models with a single denoising trajectory. In AR generation, however, multiple chunk-level trajectories are sequentially coupled through autoregressive context, making CFG reuse decisions dependent on the complete execution path. We therefore propose autoregressive trajectory-aware guidance scheduling (ATGS), which calibrates CFG refreshes over the global AR trajectory. **Third, cache reuse can induce long-term low-frequency structural loss.** Repeated approximation gradually accumulates errors in low-frequency components that encode global appearance and layout (Li et al., 2026). We therefore introduce spectral structure correction (SSC), which uses the first chunk as a stable structural reference and adaptively injects low-frequency information according to local motion, preserving global coherence while retaining dynamic details.

As shown in Fig. 1, RCS, ATGS, and SSC form a complementary framework, termed trajectory-aware reuse and adaptive correction (TRAC), for efficient and robust AR video generation. By jointly reducing redundant computation and controlling error accumulation, TRAC substantially accelerates long-video generation while maintaining high visual fidelity and temporal consistency. On VBench with SkyReels-V2, TRAC achieves a $6.03\times$ speedup over the vanilla model with only $\sim 0.6\%$ quality degradation (82.44 vs. 82.97). Compared to FlowCache-slow (Ma et al., 2026b), TRAC delivers a 0.36 higher VBench score while providing substantially better throughput ($6.03\times$ vs. $4.97\times$). Against MotionCache (Xu et al., 2026), TRAC improves the score by 0.64 while accelerating generation by nearly 29% ($6.03\times$ vs. $4.68\times$).

2 Related Work

Autoregressive Video Generation Autoregressive video generation has become a dominant paradigm for extending video duration via producing content sequentially in temporally ordered video chunks (Villegas et al., 2022; Kondratyuk et al., 2023; Henschel et al., 2025; Weng et al., 2024; Gao et al., 2024a). Recent advances in long-video diffusion, including Diffusion Forcing, CausVid, Ca2-VDM, SkyReels-V2, MAGI-1, LongLive, Self Forcing, and FramePack, further underscore the viability of autoregressive and diffusion-forcing approaches for high-fidelity long-video synthesis (Chen et al., 2024; Yin et al., 2025; Gao et al., 2024b; Chen et al., 2025; Teng et al., 2025; Yang et al., 2025; Huang et al., 2026; Zhang et al., 2026). In these models, the generation process interleaves chunk-local denoising trajectories, where later steps rely on previously generated content as historical context. Effective acceleration techniques for such a setting must therefore address both local denoising dynamics and the cumulative error that propagates across consecutive chunks (Guo et al., 2026; Ji et al., 2026; Shen et al., 2026; Lv et al., 2026; Yi et al., 2025; Luo et al., 2026; Lu et al., 2026; Ma et al., 2026b; Xu et al., 2026).

Diffusion Caching and Denoising Scheduling Training-free cache acceleration methods reduce inference cost by reusing slowly-varying intermediate features across neighboring denoising steps (Zhao et al., 2025; Lyu et al., 2025). For example, DeepCache reuses U-Net features (Ma et al.,

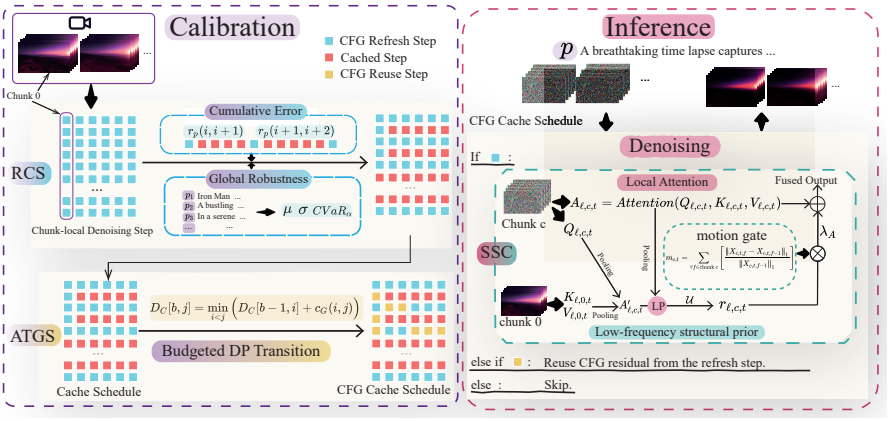

Figure 1: **Overview of the proposed trajectory-aware reuse and adaptive correction (TRAC).** TRAC comprises three complementary mechanisms for AR video generation, including robust cumulative scheduling (RCS), autoregressive trajectory-aware guidance scheduling (ATGS), and spectral structure correction (SSC).

2024b), Learning-to-Cache learns a timestep-dependent layer caching policy (Ma et al., 2024a). TeaCache estimates model-output changes from timestep-embedding-modulated inputs (Liu et al., 2025). MeanCache constructs interval average velocities from cached JVPs, builds an offline trajectory map, and determines a fixed cache schedule via graph search (Gao et al., 2026). The recent FlowCache generalizes uniform timestep caching to chunk-wise policies (Ma et al., 2026b), with a specific focus on AR video models. Moreover, to alleviate the computational cost of Classifier-free guidance (Ho & Salimans, 2022), several approaches exploit temporal redundancy in the guidance residual to reduce this overhead (Castillo et al., 2025; Lyu et al., 2025; Lee et al., 2026).

Frequency-domain Stabilization for Video Generation Extensive methods have also been proposed to address temporal consistency and structural drift in long-video generation. For instance, FreeNoise reschedules noise for extended video diffusion (Qiu et al., 2024). FreeInit refines initial noise to close the initialization gap (Wu et al., 2024). GLC-Diffusion integrates global and local denoising to balance long-range coherence and fine-grained detail (Ma et al., 2026a). Another thread tackles degradation from a frequency-domain perspective. For instance, FreeLong (Lu et al., 2024) extends video length by spectrally blending global and local temporal attention features, while FreeLong++(Lu & Yang, 2025) improves consistency across multiple frequency bands through a multi-band spectral fusion approach. Most relevant to our work, FreqForcing (Li et al., 2026) attributes drift in autoregressive long-video generation to low-frequency spectral energy, and proposes Spectral Self-Anchoring (SSA) to address it. SSA combines a local branch for motion/details, an anchor branch for low-frequency structure from early frames, and frequency-domain fusion.

3 METHOD

3.1 PRELIMINARIES AND NOTATION

Autoregressive video model. Autoregressive (AR) video models generate long videos by decomposing them into multiple fixed-length chunks and generating these chunks sequentially. Let $c \in \{0, \dots, C-1\}$ denote the chunk index and $t \in \{0, \dots, T-1\}$ denote the chunk-local denoising step. We denote the latent state of chunk c at step t as $X_{c,t}$ and the historical context from previously generated chunks as H_c. A model forward pass is formulated as:

$$v_{c,t} = F_\theta(X_{c,t}, t, \text{cond}, H_c), \tag{1}$$

where cond represents the text, image, or video conditioning.

Cache Reuse. Cache reuse (Ma et al., 2024b;a; Liu et al., 2025; Zhao et al., 2025; Ma et al., 2026b; Gao et al., 2026; Xu et al., 2026) accelerates inference by selectively executing full model forward

passes at a subset of denoising steps and reusing cached information at the remaining steps. Specifically, a cache schedule $\mathcal{S} = \{s_i\}_{i=0}^{L-1}$ specifies the denoising steps where full model forward passes are executed. Two adjacent selected steps (s_i, s_{i+1}) define a reuse span, within which intermediate steps reuse cached information. Recently, MeanCache (Gao et al., 2026) approximates intermediate velocities via cached Jacobian-vector products (JVPs), which are averaged over a subset of prompts to enable efficient video synthesis.

Classifier-free guidance. Classifier-free guidance (CFG) (Ho & Salimans, 2022) combines conditional and unconditional predictions to strengthen conditioning effects. For chunk c at denoising step t, the conditional and unconditional predictions are denoted as $\epsilon_{c,t}^{cond}$ and $\epsilon_{c,t}^{uncond}$, respectively. The guidance residual is defined as $\Delta_{c,t} = \epsilon_{c,t}^{cond} - \epsilon_{c,t}^{uncond}$, and the guided prediction is computed as $\hat{\epsilon}_{c,t} = \epsilon_{c,t}^{cond} + (w-1)\Delta_{c,t}$, where w denotes the CFG scale. To reduce overhead, CFG reuse methods (Castillo et al., 2025; Lyu et al., 2025; Lee et al., 2026) refresh $\Delta_{c,t}$ only at a subset $\mathcal{S}^C$ of denoising steps, and reuse the most recent cached residual at all other steps:

$$\Delta_{c,t} = \begin{cases} \epsilon_{c,t}^{cond} - \epsilon_{c,t}^{uncond}, & t \in \mathcal{S}^C, \\ \Delta_{c,s}, & t \notin \mathcal{S}^C,\ s = \max\{j \in \mathcal{S}^C : j < t\}, \end{cases} \tag{2}$$

where $\mathcal{S}^C$ denotes the set of steps that require a full forward pass of the model.

3.2 Trajectory-aware reuse and adaptive correction

As shown in Fig. 1, we propose trajectory-aware reuse and adaptive correction (TRAC), including robust cumulative scheduling (RCS), autoregressive trajectory-aware guidance scheduling (ATGS), and spectral structure correction (SSC), for efficient and robust AR video generation. The overall algorithm is presented in Appendix B.3.

3.2.1 Robust Cumulative Scheduling

Despite its progress, MeanCache (Gao et al., 2026) determines the cache schedule merely based on the average of local reuse errors over all chunks and prompts, while neglecting cross-chunks/prompts robustness and the cumulative impact of reuse errors along the trajectory. To address these issues, we propose robust cumulative scheduling (RCS), which formulates cache schedule selection as a risk-aware optimization problem that jointly accounts for cross-chunks/prompts robustness and cumulative rollout error.

Specifically, let a candidate schedule $\mathcal{S} = \{s_i\}_{i=0}^{L-1}$ define reuse spans $\mathcal{E}(\mathcal{S}) = \{(s_i, s_{i+1})\}_{i=0}^{L-2}$, and let $\mathcal{C}(i, i+1) = \{t : s_i < t < s_{i+1}\}$ be the cached intermediate steps in span (s_i, s_{i+1}). For calibration prompt p, chunk c, and step $t \in \mathcal{C}(i, i+1)$, the relative error between the cached prediction $\tilde{v}_{p,c,t}^{(i,i+1)}$ and the full-model prediction $v_{p,c,t}^{\text{ref}}$ is:

$$e_{p,c,t}(i, i+1) = \frac{\left\| \tilde{v}_{p,c,t}^{(i,i+1)} - v_{p,c,t}^{\text{ref}} \right\|_2}{\left\| v_{p,c,t}^{\text{ref}} \right\|_2}. \tag{3}$$

Since different chunks may exhibit substantially different sensitivities to the same reuse span, RCS uses a quantile statistic to obtain a robust chunk-aggregated error:

$$\ell_p(i, i+1, t) = Q_q\left(\{e_{p,c,t}(i, i+1) : \forall c\}\right), \tag{4}$$

where $Q_q(\cdot)$ denotes the q-th quantile. Then, RCS accumulates these robust step errors over the span and over all spans:

$$r_p(i, i+1) = \sum_{t \in \mathcal{C}(i,i+1)} \ell_p(i, i+1, t), \qquad R_p(\mathcal{S}) = \sum_{(i,i+1) \in \mathcal{E}(\mathcal{S})} r_p(i, i+1). \tag{5}$$

For cross-prompt robustness, RCS first computes the mean $\mu(\mathcal{S})$ and standard deviation $\sigma(\mathcal{S})$ of the prompt-wise errors $\{R_p(\mathcal{S}) : p \in \mathcal{D}_{\text{calib}}\}$ over the calibration set $\mathcal{D}_{\text{calib}}$, and then employs Conditional Value-at-Risk (CVaR) as the tail risk:

$$\text{CVaR}_{0.8}(\mathcal{S}) = \frac{1}{M - \lceil 0.8M \rceil + 1} \sum_{m=\lceil 0.8M \rceil}^{M} R_{(m)}(\mathcal{S}). \tag{6}$$

where $R_{(m)}$ is the m-th smallest prompt error. Under a fixed number of function evaluations (NFE) budget L, RCS minimizes:

$$J_D(\mathcal{S}) = \mu(\mathcal{S}) + \eta\sigma(\mathcal{S}) + \rho\,\mathrm{CVaR}_{0.8}(\mathcal{S}) + \lambda_q Q_{\mathrm{end}}(\mathcal{S}), \tag{7}$$

where η, ρ, λ_q are trade-off hyperparameters, and $Q_{\mathrm{end}}(\mathcal{S})$ is an endpoint-mismatch regularizer that measures the error between the final latent produced by the cached trajectory and that of the full-model trajectory. Here $\mu(\mathcal{S})$ encourages low average error, $\sigma(\mathcal{S})$ promotes robustness across prompts, and $\mathrm{CVaR}_{0.8}$ explicitly penalizes the average risk over the worst 0.2-fraction of prompts, thereby improving robustness to difficult inputs. The last term Q_{end} preserves the alignment between the final latent of the cached trajectory and that of the full-model trajectory. By solving Eq. 7 using dynamic programming, we can obtain the schedule $\mathcal{S}_{\mathrm{base}}$. Details are provided in Appendix B.1.

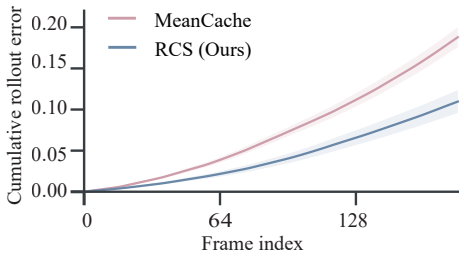

Figure 2: Cumulative rollout error (vertical) versus frame index (horizontal), with error accumulated from frame-wise output RMSE. Curves compare MeanCache and our RCS, averaged over 30 prompts randomly sampled from VBench using SkyReels-V2-1.3B-540P. Shaded regions indicate the standard error of the mean, and vanilla outputs serve as the reference.

Moreover, the first chunk plays a unique role in AR generation since it is repeatedly reused as context for subsequent chunks. Consequently, its errors can affect more subsequent chunks. Thus, for the first chunk, RCS allocates an additional NFE budget L^f and obtains a denser schedule $\mathcal{S}_{\mathrm{first}}$ by optimizing the same objective under this increased budget. This design allows aggressive reuse for later chunks while providing stronger protection for the first chunk, whose approximation errors can have the largest downstream impact. With the above designs, as shown in Fig. 2, RCS achieves a lower cumulative rollout error than MeanCache (Gao et al., 2026), whose error is higher and grows faster along the generation trajectory.

3.2.2 Autoregressive Trajectory-Aware Guidance Scheduling

Classifier-free guidance (CFG) introduces significant computational cost from repeated residual refreshes at each denoising step (Ho & Salimans, 2022; Lyu et al., 2025; Castillo et al., 2025). Although existing CFG reuse strategies (Lyu et al., 2025; Lee et al., 2026) have proven effective for bidirectional full-attention video models, they are designed for a single denoising trajectory and assume global temporal consistency across the entire sequence. In AR video generation, however, each chunk follows its own denoising trajectory, and together these trajectories form a sequential path. This key difference makes independent per-chunk scheduling suboptimal, where reuse errors accumulate along the global AR path, and a decision that appears reasonable for one chunk can lead to significant cumulative error over the full path. Therefore, a globally calibrated reuse schedule covering the entire AR trajectory is essential. Motivated by this observation, we propose autoregressive trajectory-aware guidance scheduling (ATGS), which formulates CFG reuse scheduling as a path-planning problem over the sequential AR denoising path. As shown in the purple lines in Fig. 3, CFG residual error is generally small across most steps but exhibits pronounced spikes at a few critical positions. This empirical pattern suggests that residuals can be safely reused within locally smooth intervals, while refreshes must be placed where residual changes are substantial. Consider two denoising steps $i < t$, where the CFG residual is refreshed at step i and reused at a later step t. Let $\Delta_{p,i} \in \mathbb{R}^D$

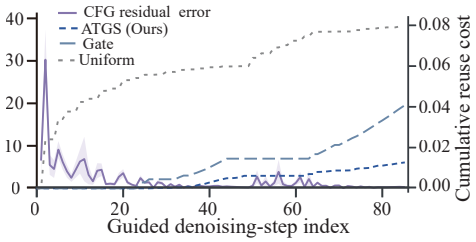

Figure 3: The horizontal axis shows the guided denoising step index along the global AR path. The left and right vertical axes plot the CFG residual error (purple) and cumulative reuse cost, respectively. ATGS (Ours), Gate, and Uniform are three CFG refresh schedules compared under the same budget. Results are on SkyReels-V2-1.3B-540P with 30 prompts from VBench.

and $\Delta_{p,t} \in \mathbb{R}^D$ denote the CFG residual for calibration prompt p at step i and t, respectively. The relative prediction error introduced by this reuse is:

$$\hat{\epsilon}^{\text{reuse}}_{p,i\to t} - \hat{\epsilon}^{\text{full}}_{p,t} = (w-1)(\Delta_{p,i} - \Delta_{p,t}), \tag{8}$$

where w is the fixed guidance scale. Thus, the residual error $\|\Delta_{p,i} - \Delta_{p,t}\|$ directly governs the approximation error introduced by reuse. To quantify this error across both chunks and prompts, ATGS first defines a metric $\delta_p(i,t)$ that captures both the magnitude and directional change between steps i and t:

$$\delta_p(i,t) = \frac{\|d_{p,t}\|_2^2 + \|d_{p,i}\|_2^2 - 2\|d_{p,t}\|_2\|d_{p,i}\|_2 \cos(d_{p,t}, d_{p,i})}{D}, \tag{9}$$

where $d_{p,n} = \text{Flatten}(\Delta_{p,n}) \in \mathbb{R}^D$. Thus, for each prompt p, the cumulative error of reusing a residual refreshed at step i until the next refreshed step j is $c_p(i,j) = \sum_{t=i+1}^{j-1} \delta_p(i,t)$. To obtain a schedule that generalizes across prompts, ATGS aggregates these errors over the calibration set $\mathcal{D}_{\text{calib}}$:

$$c_G(i,j) = \frac{1}{|\mathcal{D}_{\text{calib}}|} \sum_{p\in\mathcal{D}_{\text{calib}}} c_p(i,j). \tag{10}$$

Given the NFE budget L^C, ATGS seeks a CFG cache schedule $\mathcal{S}^C$ by selecting L^C steps $\mathcal{S}^C = \{s_i^C\}_{i=0}^{L^C-1}$ that minimize the total path error:

$$\mathcal{S}^C = \arg\min_{\mathcal{S}\in\Omega_{L^C}} \sum_{i=0}^{L^C-1} c_G(s_i^C, s_{i+1}^C), \tag{11}$$

where Ω_{L^C} denotes the set of all feasible schedules consisting of L^C refresh steps along the execution path. This optimization can be solved by standard budgeted dynamic programming in $O(L^C N^2)$ time. Details are provided in Appendix B.2.

During inference, the schedule $\mathcal{S}^C$ is applied deterministically. At each guided step t, if $t \in \mathcal{S}^C$, the CFG residual is refreshed by performing the full model forward pass. Otherwise, the residual from the most recent refresh step $i(t) = \max\{s \in \mathcal{S}^C : s \le t\}$ is reused:

$$\hat{\epsilon}_t = \epsilon_t^c + (w-1)\Delta_{i(t)}. \tag{12}$$

This formulation ensures that CFG refresh points are placed precisely at high-error positions along the global AR path, yielding a computationally efficient yet fidelity-preserving approximation of full CFG. As illustrated in Fig. 3, ATGS achieves a substantially lower cumulative reuse cost than both Gate and Uniform methods (Lyu et al., 2025; Lee et al., 2026; Castillo et al., 2025), demonstrating its effectiveness.

3.2.3 Spectral Structure Correction

Cache reuse in AR video generation accumulates low-frequency error (Li et al., 2026), which represents global appearance and layout, and becomes the dominant error source. FreqForcing (Li et al., 2026) addresses this with an auxiliary branch for a low-frequency structural prior, but incurs non-negligible attention overhead and uniformly applies this prior across all regions, potentially over-constraining high-frequency areas and obscuring local motion or fine-grained details (Ma et al., 2026a; Lu & Yang, 2025).

Motivated by these limitations, we propose spectral structure correction (SSC), which stabilizes low-frequency structures while preserving the motion and fine details of the current generation chunk at inference. Specifically, SSC employs the first chunk as a stable structural prior, serving as a reference for low-frequency information. For chunk c at layer ℓ and denoising step t, let $Q_{\ell,c,t}$ denote the query of the self-attention layer and $A_{\ell,c,t}$ denote its original output. To capture only the coarse low-frequency cues while reducing computational cost, SSC introduces an auxiliary low-resolution attention path that reuses the key and value caches ($K_{\ell,0,t}$ and $V_{\ell,0,t}$) from the first chunk ($c = 0$):

$$A'_{\ell,c,t} = \text{Attn}\left(\mathcal{P}(Q_{\ell,c,t}), \mathcal{P}(K_{\ell,0,t}), \mathcal{P}(V_{\ell,0,t})\right), \tag{13}$$

where $\mathcal{P}$ denotes a 4×4 pooling operator. This pooled attention captures the coarse relationship between the current query and the prior context while suppressing token-level fluctuations. Rather than replacing the original output, SSC extracts only the low-frequency discrepancy between the structural reference and the current output via a low-pass operator LP:

$$r_{\ell,c,t} = \mathcal{U}\left(\mathrm{LP}\left(A'_{\ell,c,t} - \mathcal{P}(A_{\ell,c,t})\right)\right), \tag{14}$$

where $\mathcal{U}$ is the upsampling operator that upsamples the pooled representation to the original resolution. Here, $r_{\ell,c,t}$ constitutes the coarse low-frequency structural correction suggested by the first chunk. Since high-frequency components are removed before fusion, fine-grained texture details and short-range motion remain governed by the original output $A_{\ell,c,t}$.

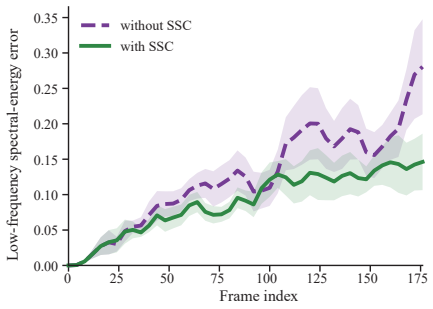

Figure 4: Low-frequency spectral-energy error (normalized by the first frame). Purple: cache-based rollout w/o SSC; green: same rollout with SSC enabled. SSC reduces low-frequency error growth under identical rollout settings. Results on SkyReels-V2-1.3B-540P with 30 VBench prompts.

To adaptively apply the low-frequency correction, SSC further introduces an adaptive correction ratio $R_{c,t} \in [R_{\min}, 1]$. We first compute a motion score $m_{c,t}$ by aggregating all adjacent-frame variations within the chunk:

$$m_{c,t} = \sum_{\forall f \in \text{chunk } c} \left[\frac{\|X_{c,t,f} - X_{c,t,f-1}\|_1}{\|X_{c,t,f-1}\|_1} \right], \tag{15}$$

where $X_{c,t,f}$ denotes the latent representation of the f-th frame in chunk c at step t. This score serves as a robust proxy for local motion intensity. The correction ratio is then obtained by mapping this score between two fixed thresholds:

$$R_{c,t} = \mathrm{Clip}\left(1 - (1 - R_{\min})\,\mathrm{Clip}\left(\frac{m_{c,t} - \tau_{\text{low}}}{\tau_{\text{high}} - \tau_{\text{low}}}, 0, 1\right), R_{\min}, 1\right), \tag{16}$$

where τ_{low} and τ_{high} are the lower and upper thresholds, and $R_{\min}$ is the minimum correction ratio. Eq. 16 ensures that chunks with low motion receive a stronger injection of low-frequency information from the first chunk, whereas chunks with high motion retain only a minimal fraction of such correction. The final corrected self-attention output is thus:

$$\bar{A}_{\ell,c,t} = A_{\ell,c,t} + \lambda_A\,\mathrm{Clip}_{\rho_A}\left(r_{\ell,c,t} \odot R_{c,t}, -\rho_A, \rho_A\right), \tag{17}$$

where λ_A is a scaling factor that controls the overall correction strength, and ρ_A is a clipping bound that prevents excessively large updates. This design enables SSC to adaptively enhance structural coherence in static or slow-moving regions while preserving the high-frequency vitality of dynamic areas. Fig. 4 shows that low-frequency error grows faster without SSC. Under the same cache-based setting, SSC substantially slows this accumulation. Note that SSC operates orthogonally to the caching schedules established by RCS and ATGS. Specifically, RCS continues to dictate which denoising steps execute full model evaluations, while ATGS governs when to refresh CFG residuals.

4 Experiments

Base models. To evaluate the efficacy of the proposed method, we select two representative autoregressive video generation backbones: SkyReels-V2-1.3B-540P and FramePack-F1 (Chen et al., 2025; Zhang et al., 2026). SkyReels-V2 adopts a hierarchical autoregressive denoising pipeline: each video section is partitioned into chunks, and earlier chunks are denoised for a predefined number of steps before subsequent chunks enter the denoising process. This execution pattern produces non-uniform denoising levels across chunks at the same sampling iteration. FramePack-F1 follows a forward-only next-frame-section prediction pipeline. It packs previous frame contexts into a fixed context budget and generates later frame sections sequentially. In our experiments, FramePack-F1 uses 25 denoising steps per generated section, 145 generated frames, and 608×640 resolution.

Baselines and Configuration. We compare with three training-free cache acceleration baselines: TeaCache, FlowCache, and MotionCache (Liu et al., 2025; Ma et al., 2026b; Xu et al., 2026). All methods use matched prompts, seeds, video lengths, resolutions, and input conditions within each backbone. The detailed hyperparameter settings are provided in Appendix A.

Table 1: Comparison on SkyReels-V2-1.3B-540P (Chen et al., 2025).

Method	Latency (s)↓	Speedup↑	VBench↑	PSNR↑	SSIM↑	LPIPS↓
Vanilla	1608.6470	1.0000×	82.9749	–	–	–
TeaCache-slow	814.0000	1.9762×	81.6700	18.9600	0.6501	**0.2472**
TeaCache-fast	686.0000	2.3450×	80.0600	18.3900	0.6121	0.3063
FlowCache-slow	323.4470	4.9734×	82.0870	17.2600	0.6370	0.3145
FlowCache-fast	268.6330	5.9883×	81.9805	16.2437	0.6088	0.3597
MotionCache	343.3827	4.6847×	81.8000	17.1731	0.6301	0.3267
TRAC	**266.8501**	**6.0283**×	**82.4440**	**20.1208**	**0.6882**	0.2568

Table 2: Comparison on FramePack-F1 (Zhang et al., 2026).

Method	Latency (s)↓	Speedup↑	Imaging Quality↑	Quality↑	I2V↑	Total↑
Vanilla	360.7360	1.0000×	0.7055	0.8203	0.9565	0.8884
TeaCache-fast	228.8670	1.5760×	**0.7108**	0.8125	0.9557	0.8841
TeaCache-slow	254.3220	1.4180×	0.7048	0.8123	0.9580	0.8851
FlowCache-fast	241.3940	1.4940×	0.7073	0.8117	**0.9609**	0.8863
FlowCache-slow	280.2290	1.2870×	0.6804	0.7780	0.9030	0.8405
MotionCache	253.9000	1.4210×	0.7049	0.8077	0.9582	0.8830
TRAC	**216.2220**	**1.6680**×	0.7046	**0.8205**	0.9560	**0.8882**

Table 3: Additional baseline comparison on SkyReels-V2.

Method	Latency (s)↓	Speedup↑	Quality↑	Semantic↑	Total↑
FreqForcing	583.4457	2.7571×	**83.4603**	**82.5981**	**83.2879**
FasterCache	458.6105	3.5077×	80.8739	77.8126	80.2617
MeanCache	405.6705	3.9654×	67.2080	61.6877	66.1040
TRAC	**266.8461**	**6.0284**×	82.9253	82.1763	82.7755

Evaluation metrics. For SkyReels-V2, we report VBench score on the full 946-prompt benchmark together with PSNR, SSIM, and LPIPS diagnostics. For FramePack-F1, we report the VBench++ Imaging Quality, Quality, I2V, and Total scores. Latency is measured end-to-end. We also report the speedup relative to Vanilla model inference.

4.1 Results

Main Results. As shown in Tables 1 and 2, across SkyReels-V2 and FramePack-F1, TRAC consistently provides a strong efficiency–quality trade-off. On SkyReels-V2, TRAC reduces dense latency from 1608.6470 seconds to 266.8501 seconds, an 83.4% reduction corresponding to a 6.0283× speedup. Its VBench score is 82.4440, only about 0.6% lower than Vanilla. Among compared methods, TRAC achieves the highest VBench score, a PSNR of 20.1208, and an SSIM of 0.6882. Its LPIPS of 0.2568 is second only to TeaCache-slow, while requiring substantially lower latency. Compared with FlowCache-fast at a similar latency, TRAC improves VBench score from 81.9805 to 82.4440, PSNR from 16.2437 to 20.1208, SSIM from 0.6088 to 0.6882, and LPIPS from 0.3597 to 0.2568, showing significant performance advantages.

On FramePack-F1, under the fixed FramePack configuration with $\mathrm{cfg} = 1$, seed 1024, TRAC reduces the latency of dense Vanilla inference from 360.7360 to 216.2220 seconds, yielding the lowest latency and the highest speedup of 1.6680×. It also achieves the highest Quality score of 0.8205. FlowCache-fast obtains the highest I2V score of 0.9609, while Vanilla obtains the highest Total score of 0.8884. TRAC reaches 0.8882, only 0.0002 lower. TeaCache-fast obtains the highest Imaging Quality score of 0.7108. Compared with TeaCache-slow, FlowCache-fast, and MotionCache, TRAC is faster and obtains higher Total scores, while maintaining a comparable I2V score. These results show that TRAC reaches the most efficient operating point while preserving aggregate quality close to dense Vanilla inference.

Additional Baseline Comparison. To determine if the efficiency-quality behavior of TRAC holds across a broader range of acceleration mechanisms, we further evaluate it against FreqForcing, FasterCache, and MeanCache on SkyReels-V2. These selected baselines cover various complementary training-free strategies, such as spectral correction, cache reuse, and JVP-based schedule selec-

tion. Since full evaluation of every auxiliary baseline is computationally expensive, we randomly sample 94 prompts from the VBench prompt set and evaluate all methods under matched prompts, seeds, resolutions, and video lengths. Table 3 presents the comparison under a matched evaluation setting. TRAC significantly outperforms the other methods in efficiency, achieving a $6.03\times$ speedup with a latency of only 266.8s. Although FreqForcing obtains slightly higher VBench aggregates (e.g., a Total score of 83.29 vs. TRAC's 82.78), it requires a substantial 583.4s to complete. Both FasterCache and MeanCache fall behind TRAC in terms of speed and generation metrics. Consequently, TRAC achieves a superior trade-off, successfully preserving high-fidelity visual quality while providing the most aggressive speedup. In Appendix C, we provide quality comparisons.

4.2 Ablation Study

As shown in Fig. 5, we conduct the component ablation on the SkyReels-V2-1.3B-540P, using VBench score as the quality metric. The full TRAC setting combines RCS, ATGS, and SSC. Removing SSC reduces latency from 266.9 to 259.1 seconds, but lowers VBench from 82.44 to 81.81, indicating that SSC mainly contributes to quality. Removing ATGS increases latency to 291.5 seconds while maintaining a similar VBench score of 82.39, demonstrating the efficiency contribution of CFG residual reuse. Removing RCS increases latency to 572.9 seconds, more than doubling that of TRAC, despite a VBench score of 83.24. These comparisons highlight the complementary roles of RCS, ATGS, and SSC. Detailed sensitivity analyses of the SSC correction strength λ_A and the RCS objective weights η, ρ, and λ_q are reported in Appendix D.

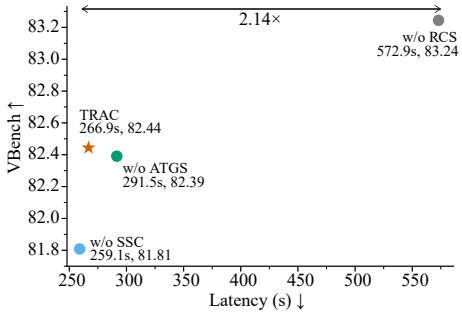

Figure 5: **Component ablation on SkyReels-V2.** The scatter plot compares latency and VBench total after removing one component at a time.

5 Limitations and Future Work

First, while TRAC is effective on two widely used autoregressive backbones and achieves the best trade-off between performance and speed, its generalization to emerging AR video models remains an open question. Addressing this would require considerable computational resources, which are not currently available to us. Second, although TRAC already outperforms existing caching methods in terms of speedup, it is orthogonal to complementary acceleration techniques, including quantization, distillation, and sparse attention. Integrating TRAC with these techniques may yield additional efficiency gains, a direction we leave for future work.

6 Conclusion

In this paper, we present trajectory-aware reuse and adaptive correction (TRAC), a training-free framework for efficient autoregressive (AR) video generation. Existing acceleration methods mainly target single-trajectory generation with bidirectional attention. AR video generation, by contrast, sequentially couples chunk-level denoising trajectories. Consequently, approximation errors accumulate and propagate through the generation process. TRAC addresses this challenge with three components, including robust cumulative scheduling (RCS), autoregressive trajectory-aware guidance scheduling (ATGS), and spectral structure correction (SSC). RCS selects cache reuse schedules by cumulative rollout error and cross-chunk/prompt variation. ATGS coordinates CFG refreshes along the global AR trajectory. SSC restores low-frequency structure of the first chunk to correct long-term structural loss. Experiments on SkyReels-V2 and FramePack-F1 show that, compared with existing methods, TRAC achieves both the highest inference efficiency and the best generation quality for AR video generation.

AI USE STATEMENT

During the preparation of this manuscript, we used generative AI tools solely for language polishing and code-writing assistance. Generative AI tools were not used to replace experimental execution, generate experimental data, or serve as a source of unverified research conclusions or literature evidence. The authors have rigorously checked and reviewed all generated content to ensure its correctness, and they take full responsibility for the final code, content, and claims presented in this work.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the anonymized source code in the supplementary material. The method and experimental settings are described in Sections 3 and 4, with additional implementation details provided in Appendix A. Crucially, all proposed schedules (the shared denoising path, first-chunk schedule, and CFG refresh set) are fixed post-calibration, ensuring deterministic execution with zero online scoring overhead. All experiments use fixed random seeds and are evaluated under strictly identical conditions (e.g., prompts, resolutions, and video lengths).

REFERENCES

Angela Castillo, Jonas Kohler, Juan C Pérez, Juan Pablo Pérez, Albert Pumarola, Bernard Ghanem, Pablo Arbeláez, and Ali Thabet. Adaptive guidance: Training-free acceleration of conditional diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1962–1970, 2025.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, et al. Skyreels-v2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

Huanlin Gao, Ping Chen, Fuyuan Shi, Ruijia Wu, Li YanTao, Qiang Hui, Ting Lu, Chao Tan, Shaoan Zhao, Zhaoxiang Liu, et al. Meancache: From instantaneous to average velocity for accelerating flow matching inference. In *International Conference on Learning Representations*, volume 2026, pp. 60466–60485, 2026.

Kaifeng Gao, Jiaxin Shi, Hanwang Zhang, Chunping Wang, and Jun Xiao. Vid-gpt: Introducing gpt-style autoregressive generation in video diffusion models. *arXiv preprint arXiv:2406.10981*, 2024a.

Kaifeng Gao, Jiaxin Shi, Hanwang Zhang, Chunping Wang, Jun Xiao, and Long Chen. Ca2-vdm: Efficient autoregressive video diffusion model with causal generation and cache sharing. *arXiv preprint arXiv:2411.16375*, 2024b.

Hang Guo, Zhaoyang Jia, Jiahao Li, Bin Li, Yuanhao Cai, Jiangshan Wang, Yawei Li, and Yan Lu. Efficient autoregressive video diffusion with dummy head. *arXiv preprint arXiv:2601.20499*, 2026.

Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2568–2577. IEEE, 2025.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

Yicheng Ji, Zhizhou Zhong, Jun Zhang, Qin Yang, XiTai Jin, Ying Qin, Wenhan Luo, Shuiyang Mao, Wei Liu, and Huan Li. Forcing-kv: Hybrid kv cache compression for efficient autoregressive video diffusion models. *arXiv preprint arXiv:2605.09681*, 2026.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15240–15252. IEEE, 2025.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

Yunghee Lee, Byeonghyun Pak, Junwha Hong, and Hoseong Kim. Tortoise and hare guidance: Accelerating diffusion model inference with multirate integration. *Advances in Neural Information Processing Systems*, 38:2871–2898, 2026.

Jiatong Li, Leo Liang, Linghe Kong, and Yulun Zhang. Freqforcing: Autoregressive long video generation via spectral self-anchoring. *arXiv preprint arXiv:2607.27110*, 2026.

Enhuai Liu, Yunke Wang, Changming Sun, and Chang Xu. D2cache: Second-order delta caching for higher video diffusion acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43589–43599, 2026.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025. URL https://arxiv.org/abs/2411.19108.

Yu Lu and Yi Yang. Freelong++: Training-free long video generation via multi-band spectralfusion. *arXiv preprint arXiv:2507.00162*, 2025.

Yu Lu, Yuanzhi Liang, Linchao Zhu, and Yi Yang. Freelong: Training-free long video generation with spectralblend temporal attention. *Advances in Neural Information Processing Systems*, 37: 131434–131455, 2024.

Yu Lu, Junjie Yang, Piotr Koniusz, YuXin Song, and Yi Yang. Fademem: Distance-aware memory consolidation for autoregressive video diffusion. *arXiv preprint arXiv:2606.10671*, 2026.

Jiayi Luo, Qiyan Liu, Tengyang Wang, JunHao Liu, Jiayu Chen, Cong Wang, Hanxin Zhu, Chen Gao, Xiaobin Hu, Qingyun Sun, et al. Future forcing: Future-aware training-free kv cache policy for autoregressive video generation. *arXiv preprint arXiv:2605.30083*, 2026.

Chengtao Lv, Yumeng Shi, Yushi Huang, Ruihao Gong, Shen Ren, and Wenya Wang. Light forcing: Accelerating autoregressive video diffusion via sparse attention. *arXiv preprint arXiv:2602.04789*, 2026.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, volume 2025, pp. 33132–33156, 2025.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *Advances in Neural Information Processing Systems*, 37: 133282–133304, 2024a.

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

Yongjia Ma, Junlin Chen, Donglin Di, Qi Xie, Lei Fan, Wei Chen, Na Zhao, and Xun Yang. Tuning-free long video generation via global-local collaborative diffusion. *ACM Transactions on Multimedia Computing, Communications and Applications*, 22(7):1–25, 2026a.

Yuexiao Ma, Xuzhe Zheng, Jing Xu, Xiwei Xu, Feng Ling, Xiawu Zheng, Huafeng Kuang, Huixia Li, Xing Wang, Xuefeng Xiao, et al. Flow caching for autoregressive video generation. *arXiv preprint arXiv:2602.10825*, 2026b.

Haonan Qiu, Menghan Xia, Yong Zhang, Yingqing He, Xintao Wang, Ying Shan, and Ziwei Liu. Freenoise: Tuning-free longer video diffusion via noise rescheduling. In *International Conference on Learning Representations*, volume 2024, pp. 5260–5274, 2024.

Jinliang Shen, Lianghao Su, Zheming Li, Kang He, ZiLiang Lai, Yanbing Jiang, and Chengru Song. Headcast: Casting attention heads for efficient autoregressive video generation. *arXiv preprint arXiv:2607.20125*, 2026.

Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, et al. Art• v: Auto-regressive text-to-video generation with diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 7395–7405. IEEE, 2024.

Tianxing Wu, Chenyang Si, Yuming Jiang, Ziqi Huang, and Ziwei Liu. Freeinit: Bridging initialization gap in video diffusion models. In *European conference on computer vision*, pp. 378–394. Springer, 2024.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Jing Xu, Yuexiao Ma, Xuzhe Zheng, Xing Wang, Shiwei Liu, Chenqian Yan, Xiawu Zheng, Rongrong Ji, Fei Chao, and Songwei Liu. Motion-aware caching for efficient autoregressive video generation. *arXiv preprint arXiv:2605.01725*, 2026.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025.

Jung Yi, Wooseok Jang, Paul Hyunbin Cho, Jisu Nam, Heeji Yoon, and Seungryong Kim. Deep forcing: Training-free long video generation with deep sink and participative compression. *arXiv preprint arXiv:2512.05081*, 2025.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974. IEEE, 2025.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. *Advances in Neural Information Processing Systems*, 38:30546–30566, 2026.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *International Conference on Learning Representations*, volume 2025, pp. 3296–3319, 2025.

Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching. *arXiv preprint arXiv:2507.02860*, 2025.

A Implementation Details

All experiments are implemented in PyTorch and executed on NVIDIA A800 GPUs with batch size 1.

SkyReels-V2. Following the official SkyReels-V2 implementation, we use 177 frames, 2 autoregressive sections with 5 causal chunks per section, 17 overlap frames, 50 denoising steps along each chunk-local trajectory, an autoregressive step offset of 5, CFG scale $w = 6.0$, shift 8.0, 24 fps, and 540P resolution. For TRAC, RCS uses $\mathcal{S}_{\text{first}}$ for the first chunk and $\mathcal{S}_{\text{base}}$ for subsequent chunks. The first and subsequent chunks retain 30 and 10 full-model forward passes, respectively. The RCS objective uses $q = 0.90$ and $\text{CVaR}_{0.8}$, with $\eta = 0.03$, $\rho = 0.01$, and $\lambda_q = 0.003$. The reported run contains 86 conditional full-model evaluations. ATGS refreshes the CFG residual at 72 guided denoising steps and reuses the most recent cached CFG residual at the remaining 14 guided denoising steps.

FramePack-F1. Following the FramePack-F1 implementation, we use one input image, 145 generated frames, 4 generated sections, 25 denoising steps per section, a latent window size of 9, and 608×640 resolution. Following the fixed FramePack evaluation configuration, we use seed 1024, $\text{cfg} = 1$, a distilled guidance scale of $\text{gs} = 10$, and a CFG rescale of $\text{rs} = 0$. For TRAC, RCS retains 24 full-model forward passes in the first generated section and 10 full-model forward passes in each subsequent section. Across the four generated sections, this gives 54 full-model forward passes and 46 reuse positions. The RCS objective uses $q = 0.90$ and $\text{CVaR}_{0.8}$, with $\eta = 0.03$, $\rho = 0.01$, and $\lambda_q = 0.003$. Since $\text{cfg} = 1$, ATGS is inactive during FramePack-F1 generation, and SSC is enabled during final generation. The remaining calibration and execution settings are provided in Appendix A.

SSC configuration. For SkyReels-V2, SSC uses the first chunk as the structural reference, whereas for FramePack-F1 it uses the first generated section. Both backbones use the 4×4 pooling operator $\mathcal{P}$, $\lambda_A = 0.40$, $\rho_A = 0.25$, $\tau_{\text{low}} = 0.08$, $\tau_{\text{high}} = 0.20$, and $R_{\min} = 0.25$. The remaining implementation-specific SSC settings, including the attention-layer range, anchor-chunk configuration, smoothing kernel, and γ, are provided in Appendix A.

FramePack-F1 RCS calibration. For FramePack-F1, RCS calibration uses an outer JVP-based scheduler, with state-based JVP prediction at reuse positions. The observer uses a three-position warm-up and evaluates candidate jumps of up to eight positions. Schedules are selected according to the relative prediction error in the ℓ_2 norm. The selected first-section and later-section schedules are:

$$\mathcal{S}_{\text{first}} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24\},$$

$$\mathcal{S}_{\text{base}} = \{0, 1, 2, 3, 4, 6, 8, 9, 11, 13\}.$$

The first-section schedule uses the forced prefix `[0,1,2,3]`. At RCS reuse positions, the transformer is not executed, and the SSC correction branch is not invoked.

SSC implementation. The SSC correction branch is applied to attention layers 22–31. The low-resolution correction path uses one anchor chunk, with correction starting from chunk 1, and applies a 3×3 smoothing kernel. The motion-dependent correction ratio uses $\gamma = 1.0$. The correction is applied on the conditional prediction branch and does not introduce an additional unconditional model evaluation.

B Dynamic Programming Details for Budgeted Schedule Search

This appendix details the dynamic programming (DP) algorithms utilized to solve the budgeted schedule search problems formulated in Sections 3.2 and 3.3.

B.1 RCS Dynamic Programming

To determine the optimal retained denoising steps under a given budget, RCS precomputes the schedule score for all feasible reuse spans across the calibration set. After calibration, each feasible span is assigned an aggregated scalar score, denoted as $a_D(i, j)$, which is computed from the RCS span statistics described in Section 3.2.

State Definition. Let $D_D[b, j]$ denote the minimum accumulated RCS score of a partial schedule that selects exactly b retained steps and terminates at step j. An auxiliary table $\mathrm{Prev}_D[b, j]$ records the optimal predecessor to facilitate trajectory backtracking.

Initialization. In practice, the DP can be initialized from the fixed prefix used by the implementation. Assuming this warm-up phase ends at step w, we initialize the DP at this prefix boundary:

$$D_D[w + 1, w] = 0, \tag{18}$$

with all other states initialized to $+\infty$.

State Transition. For each valid candidate step j and its predecessor $i < j$, the state transition is formulated as:

$$D_D[b, j] = \min_{i<j} \left(D_D[b - 1, i] + a_D(i, j) \right). \tag{19}$$

The index i that minimizes Eq. 19 is stored in $\mathrm{Prev}_D[b, j]$.

Termination. Let T_{end} denote the terminal node of the schedule. Upon evaluating all valid paths that reach T_{end} under the NFE budget B_D, the solver selects the state $D_D[B_D, T_{\mathrm{end}}]$ yielding the minimum accumulated score. The baseline schedule $\mathcal{S}_{\mathrm{base}}$ is reconstructed by backtracking through Prev_D. The denser first-chunk schedule $\mathcal{S}_{\mathrm{first}}$ is computed using the identical solver formulation but with the expanded budget B_f.

Algorithm 1 Budgeted DP for RCS

Require: Aggregated RCS span scores $a_D(i, j)$, NFE budget B_D, warm-up end step w, terminal node T_{end}
Ensure: Optimal cache schedule $\mathcal{S}_{\mathrm{base}}$

```
Initialize D_D[b, j] ← +∞ and Prev_D[b, j] ← −1 for all valid b, j.
D_D[w + 1, w] ← 0 {Initialize from the fixed warm-up prefix}
for b = w + 2 to B_D do
    for each candidate retained step j ≤ T_end do
        for each valid predecessor i < j do
            if D_D[b − 1, i] + a_D(i, j) < D_D[b, j] then
                D_D[b, j] ← D_D[b − 1, i] + a_D(i, j)
                Prev_D[b, j] ← i
            end if
        end for
    end for
end for
S_base ← Backtrack(D_D, Prev_D, B_D, T_end)
return S_base
```

B.2 ATGS Dynamic Programming

For ATGS, the optimal schedule minimizes the total CFG residual-reuse cost. The solver frames this as a fixed-budget path-planning problem. Let the ordered CFG events be $0, \ldots, N - 1$, and use N as a terminal node. The calibrated edge score $c_G(i, j)$ evaluates the cumulative reuse error between two consecutive refresh steps i and j. Optional mandatory nodes, such as boundary events, can be enforced by masking invalid transitions.

State Definition. Let $D_C[b, j]$ denote the minimum accumulated CFG reuse cost of a path that ends at refresh step j utilizing exactly b refresh allocations. Similar to RCS, $\mathrm{Prev}_C[b, j]$ stores the previous refresh step for path reconstruction.

Initialization. The sequence originates at the initial event 0, which functions as the first refresh node:

$$D_C[1, 0] = 0, \tag{20}$$

with all other states initialized to $+\infty$.

State Transition. If the next refresh step is assigned to j, the CFG residual from the preceding refresh step i is reused across all intermediate steps. The transition is identical in structure to RCS:

$$D_C[b, j] = \min_{i<j} \left(D_C[b-1, i] + c_G(i, j) \right). \tag{21}$$

Termination. After exhausting the CFG budget L^C, the solver identifies the valid state covering the terminal node N with the minimum total cost. The final ATGS schedule $\mathcal{S}^C$ is retrieved via backtracking through Prev_C.

Algorithm 2 Budgeted DP for ATGS

Require: Calibrated CFG reuse costs $c_G(i, j)$, CFG refresh budget L^C, terminal node N
Ensure: Optimal CFG refresh schedule $\mathcal{S}^C$

```
Initialize D_C[b, j] ← +∞ and Prev_C[b, j] ← −1 for all valid b, j.
D_C[1, 0] ← 0 {Initialize the first refresh event}
for b = 2 to L^C do
    for each candidate refresh step j ≤ N do
        for each valid predecessor i < j do
            if span (i, j) satisfies constraints and D_C[b − 1, i] + c_G(i, j) < D_C[b, j] then
                D_C[b, j] ← D_C[b − 1, i] + c_G(i, j)
                Prev_C[b, j] ← i
            end if
        end for
    end for
end for
S^C ← Backtrack(D_C, Prev_C, L^C, N)
return S^C
```

B.3 Unified algorithm For TRAC

Algorithm 3 summarizes the offline calibration stage, which produces the RCS schedules $\mathcal{S}_{\text{base}}$ and $\mathcal{S}_{\text{first}}$, ATGS' $\mathcal{S}^C$, and the SSC low-frequency correction branch. Algorithm 4 describes the inference stage that applies RCS, ATGS, and SSC during chunk-wise video generation.

Algorithm 3 Offline calibration

Require: calibration prompts $\mathcal{D}_{\text{calib}}$, autoregressive generator F_θ, NFE budgets L, L^f, L^C
Ensure: cache schedules $\mathcal{S}_{\text{base}}$, $\mathcal{S}_{\text{first}}$, CFG cache schedule $\mathcal{S}^C$, fixed SSC correction branch

1. Collect the full-model predictions $v^{\text{ref}}_{p,c,t}$ and the cached predictions $\tilde{v}^{(i,i+1)}_{p,c,t}$ on chunk-local denoising trajectories.
2. Compute the RCS span errors $r_p(i, j)$ and solve RCS under the NFE budget L to obtain $\mathcal{S}_{\text{base}}$.
3. Solve RCS under the larger first-chunk NFE budget L^f to obtain the denser schedule $\mathcal{S}_{\text{first}}$.
4. Fix the execution trace induced by $\mathcal{S}_{\text{first}}$ and $\mathcal{S}_{\text{base}}$, and collect the full CFG residuals $\Delta_{p,t}$ on this trace.
5. Aggregate the residual-reuse costs $c_G(i, j)$ over the calibration set $\mathcal{D}_{\text{calib}}$ and solve the budgeted ATGS optimization under the NFE budget L^C to obtain $\mathcal{S}^C$.
6. Fix the SSC low-frequency correction branch.

Algorithm 4 Inference

Require: calibrated cache schedules $\mathcal{S}_{\text{first}}$, $\mathcal{S}_{\text{base}}$, CFG cache schedule $\mathcal{S}^C$, fixed SSC correction branch, CFG scale w
Ensure: generated latent chunks $\{X_{c,T-1}\}_{c=0}^{C-1}$

for each generated chunk c **do**
  Use $\mathcal{S}_{\text{first}}$ for the first chunk and $\mathcal{S}_{\text{base}}$ thereafter.
  for each guided denoising step t **do**
    if the current step is an RCS full-evaluation step **then**
      Compute the conditional prediction using the SSC path when enabled.
    else
      Reuse the cached or JVP-predicted conditional output and skip SSC.
    end if
    if the step is in $\mathcal{S}^C$ or has no compatible cached residual **then**
      Compute the unconditional prediction $\epsilon_{c,t}^{uncond}$ and refresh the CFG residual $\Delta_{c,t}$.
    else
      Reuse the latest compatible CFG residual $\Delta_{c,i(t)}$.
    end if
    Combine the conditional prediction with the refreshed or corrected residual: $\hat{\epsilon}_{c,t} = \epsilon_{c,t}^{cond} + (w-1)\Delta_{c,i(t)}$.
  end for
end for

C Qualitative Results

Here, following prior cache-acceleration studies (Ma et al., 2026b; Xu et al., 2026), we provide matched-prompt frame comparisons. All methods use matched prompts and seeds. Specifically, we provide eight matched-prompt qualitative comparisons on SkyReels-V2. Each panel shows four sampled frames for Vanilla, TeaCache-slow, FlowCache-slow, MotionCache, and TRAC under the same prompt and seed.

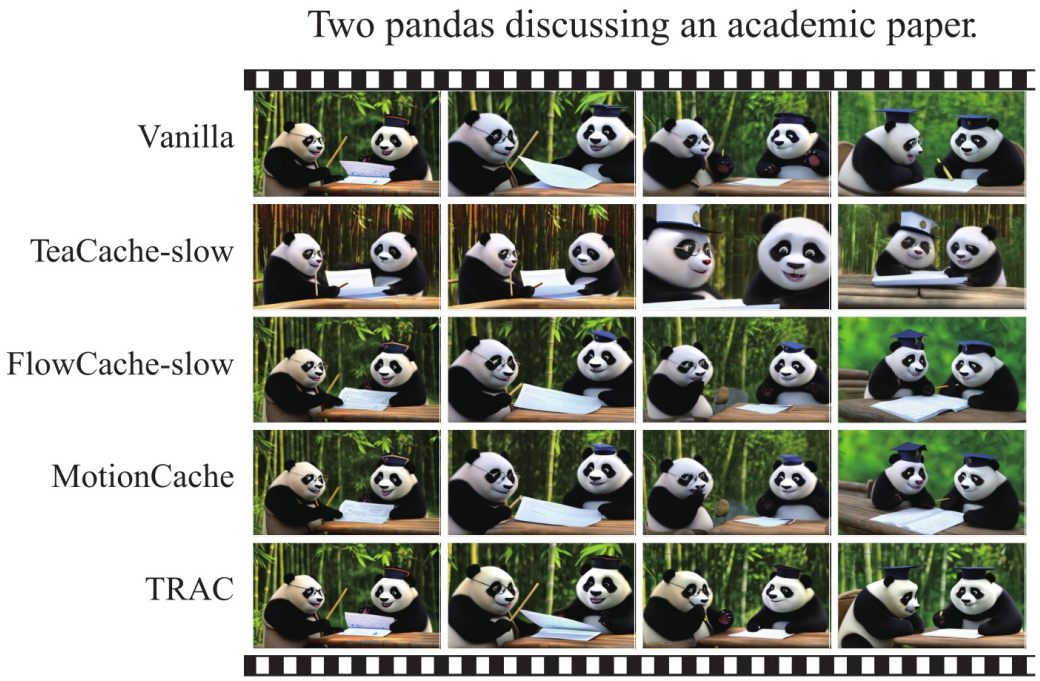

Figure 6: **Qualitative comparison on SkyReels-V2.** Prompt: "Two pandas discussing an academic paper."

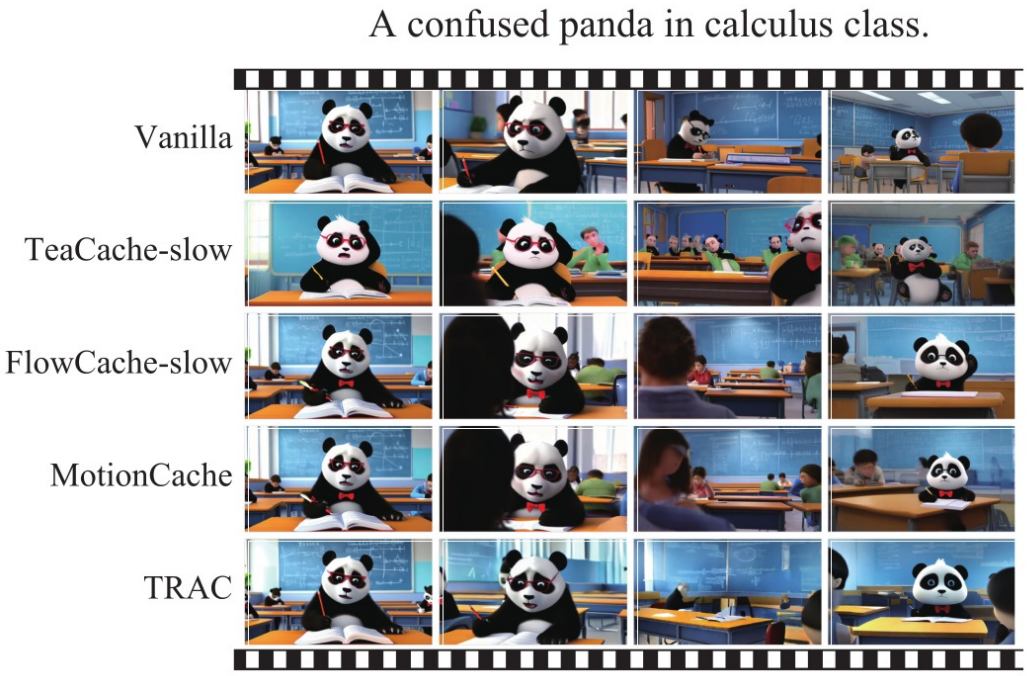

Figure 7: **Qualitative comparison on SkyReels-V2.** Prompt: "A confused panda in calculus class."

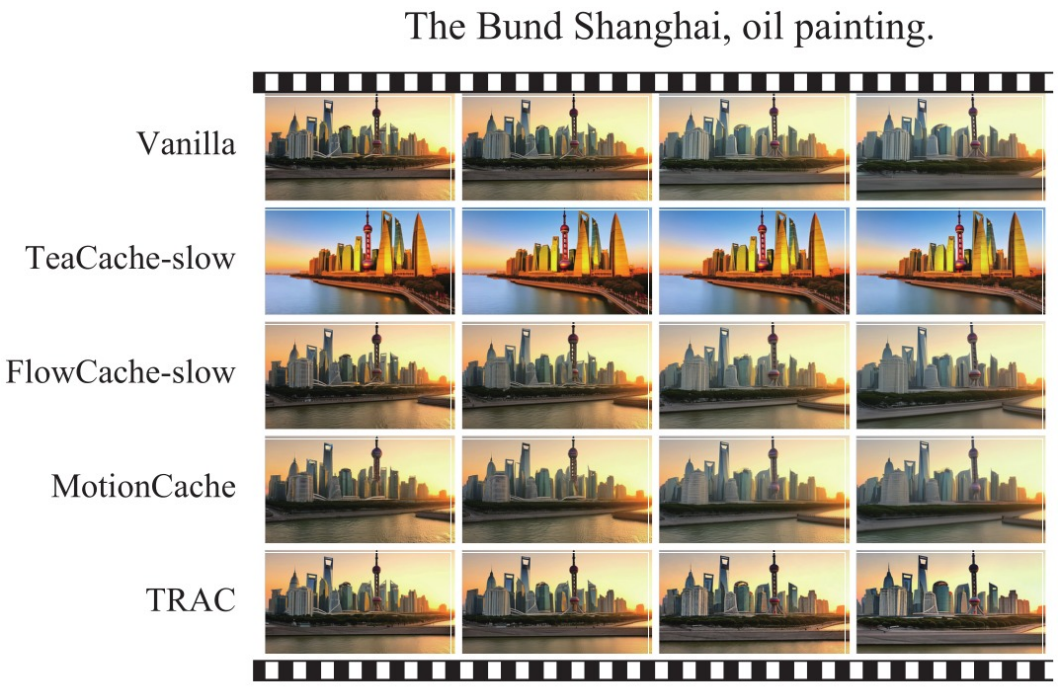

Figure 8: **Qualitative comparison on SkyReels-V2.** Prompt: "The Bund Shanghai, oil painting."

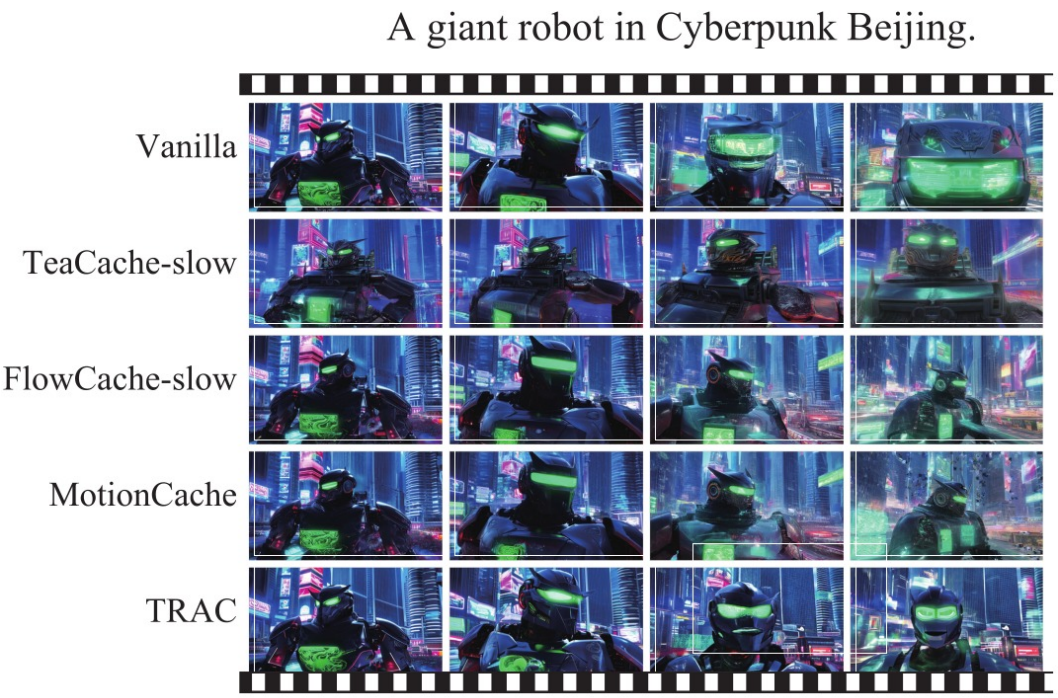

Figure 9: **Qualitative comparison on SkyReels-V2.** Prompt: "A giant robot in Cyberpunk Beijing."

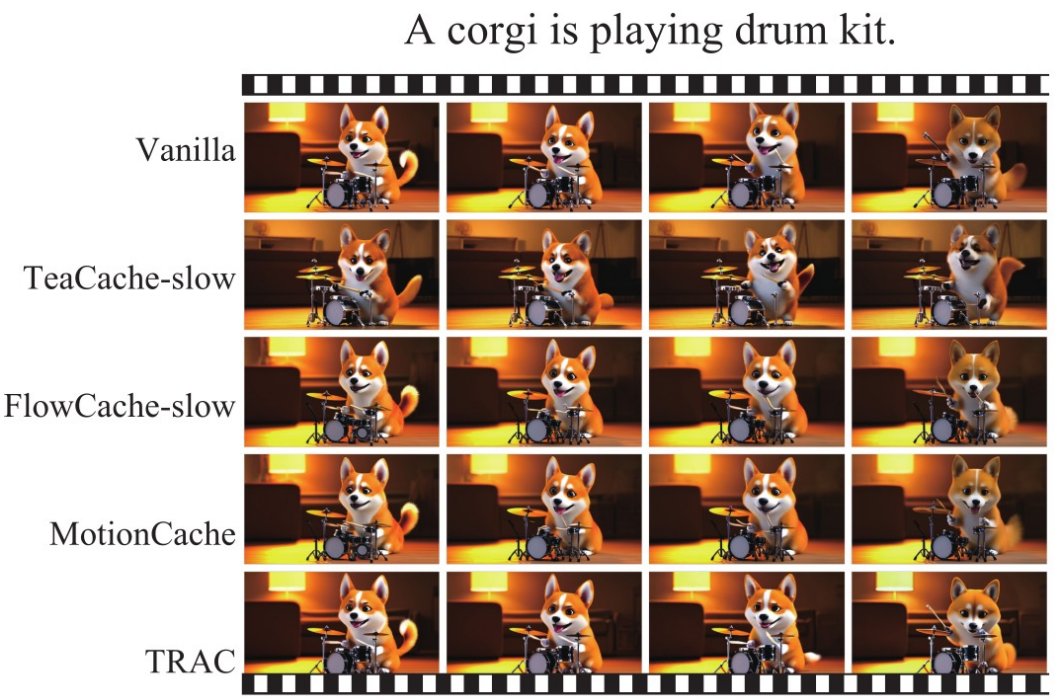

Figure 10: **Qualitative comparison on SkyReels-V2.** Prompt: "A corgi is playing drum kit."

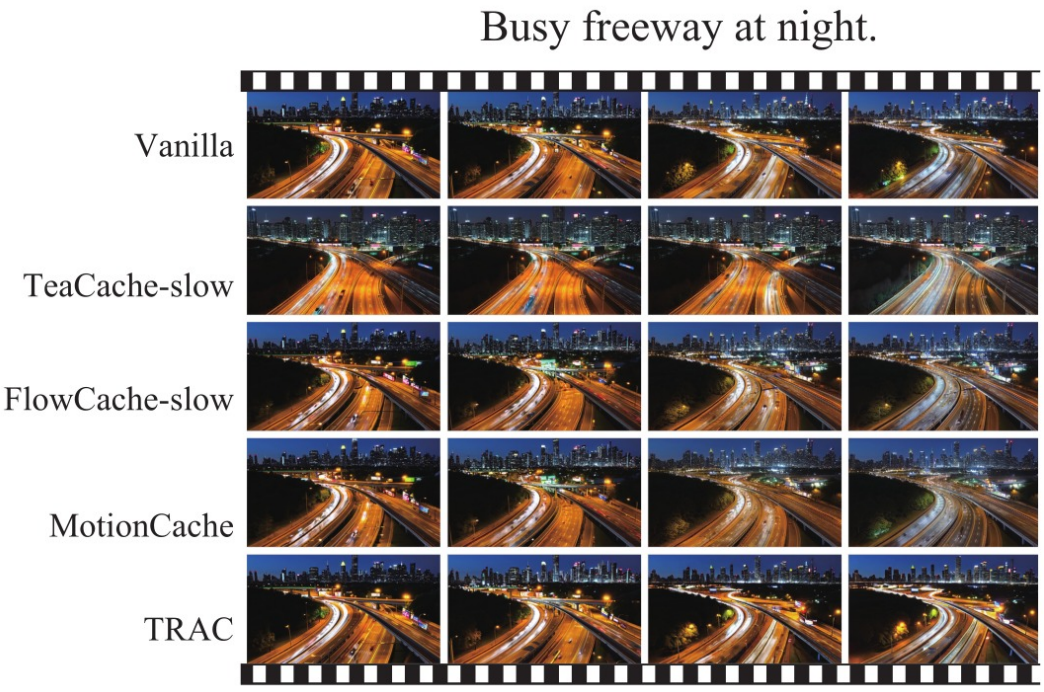

Figure 11: **Qualitative comparison on SkyReels-V2.** Prompt: "Busy freeway at night."

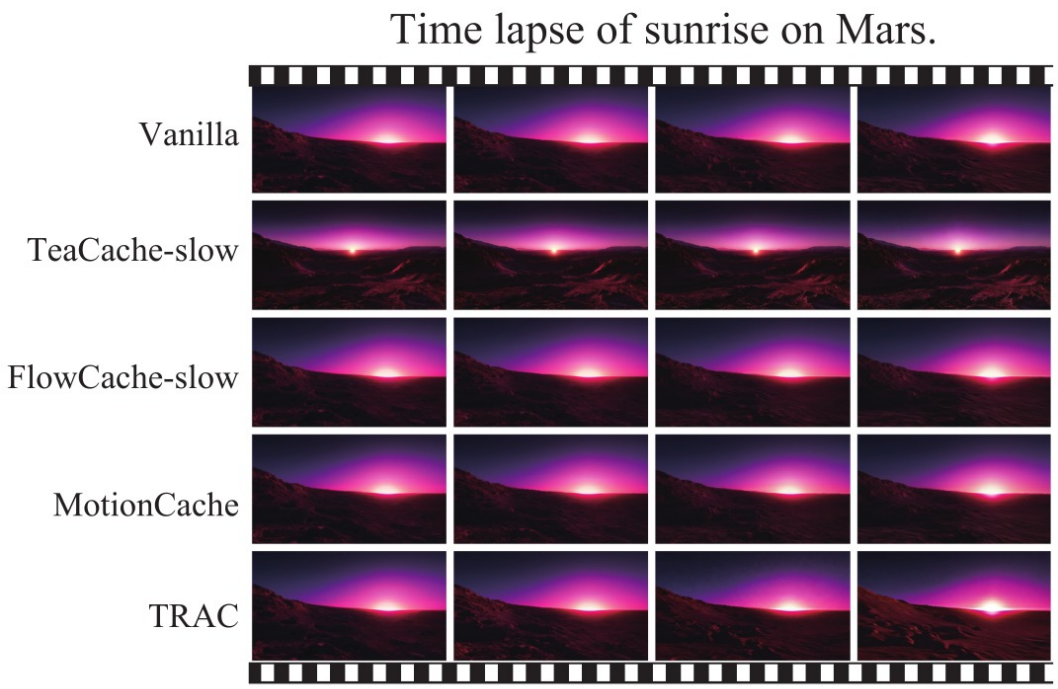

Figure 12: **Qualitative comparison on SkyReels-V2.** Prompt: "Time lapse of sunrise on Mars."

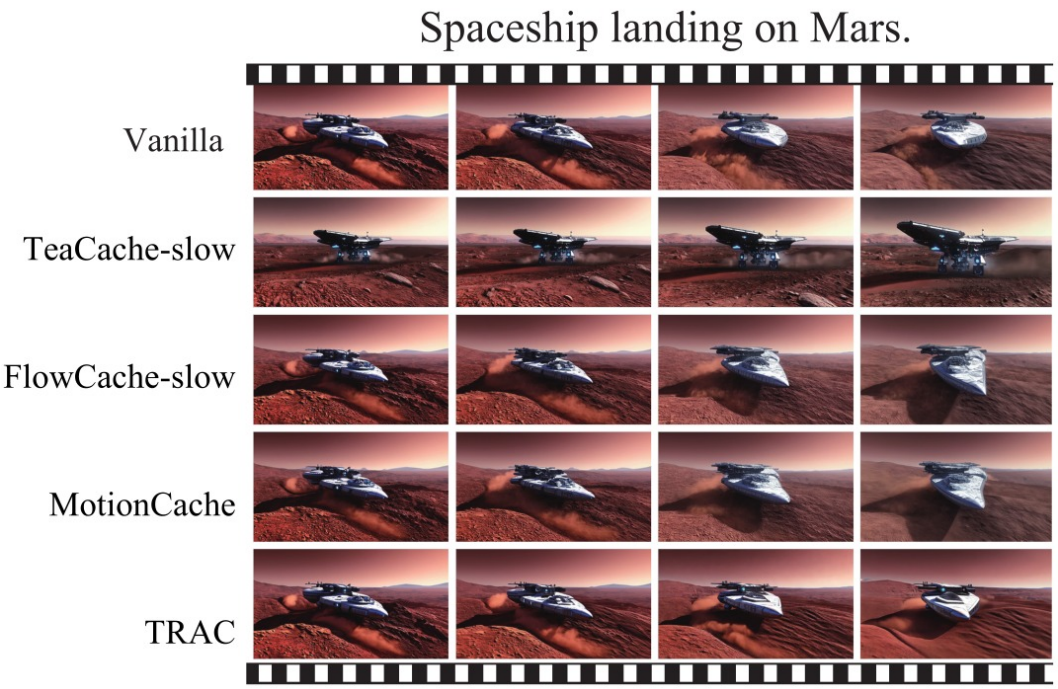

Figure 13: **Qualitative comparison on SkyReels-V2.** Prompt: "Spaceship landing on Mars."

D Additional Ablation Studies

We evaluate the sensitivity of the SSC correction strength and the RCS objective weights on the matched 94-prompt SkyReels-V2 subset. Unless otherwise stated, the remaining settings follow the default configuration. In Table 4, $\lambda_A = 0.40$ gives the highest Total score and is used in the reported experiments.

Table 4: Ablation studies on λ_A.

λ_A	Quality↑	Semantic↑	Total↑
0.20	82.4367	81.7599	82.3013
0.30	82.7358	81.7433	82.5373
0.40	**82.9253**	82.1763	**82.7755**
0.50	82.0764	82.4972	82.1606
0.60	82.0731	**83.4813**	82.3547

We next vary the three RCS objective weights independently. The other two weights are fixed to their default values in each table. Some sweep settings select the same calibrated schedule after the discrete path search, and therefore lead to identical evaluation scores.

Table 5: Sensitivity to the RCS weight η, with $\rho = 0.01$ and $\lambda_q = 0.003$ fixed.

η	Quality↑	Semantic↑	Total↑
0.03	82.9253	**82.1763**	**82.7755**
3.00	**82.9524**	81.9728	82.7565
10.00	82.9309	81.9741	82.7395

Table 6: Sensitivity to the RCS weight ρ, with $\eta = 0.03$ and $\lambda_q = 0.003$ fixed.

ρ	Quality↑	Semantic↑	Total↑
0.01	82.9253	**82.1763**	**82.7755**
3.00	**82.9524**	81.9728	82.7565
20.00	**82.9524**	81.9728	82.7565

Table 7: Sensitivity to the RCS weight λ_q, with $\eta = 0.03$ and $\rho = 0.01$ fixed.

λ_q	Quality↑	Semantic↑	Total↑
0.003	**82.9253**	82.1763	**82.7755**
0.30	82.6695	**82.7072**	82.6770
1.00	82.6695	**82.7072**	82.6770