TEMPACT: ADVANCING TEMPORAL PLAUSIBILITY IN AUTOREGRESSIVE VIDEO GENERATION VIA PLANNER-EXECUTOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

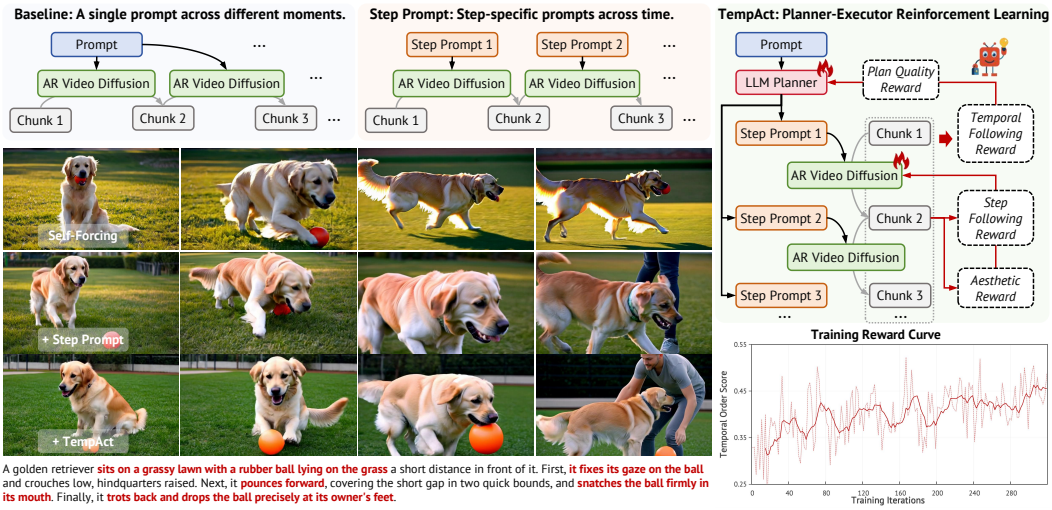


Figure 1: **Overview and Motivation of TempAct. Framework.** Single-prompt AR generation conditions every chunk on the same global instruction, while step-prompt generation provides explicit stage-wise conditions but still relies on a fixed executor. TempAct introduces a planner–executor RL framework that jointly optimizes temporal decomposition and prompt-transition execution. **Qualitative comparison**. Compared with single-prompt and step-prompt baselines, TempAct produces more faithful event progression under temporally complex instructions. **Training dynamics**. The increasing reward curve shows that hierarchical planner–executor optimization provides effective learning signals for temporal plausibility.

ABSTRACT

Autoregressive (AR) video diffusion models enable low-latency streaming generation by synthesizing videos chunk by chunk with cached visual context, but this chunk-wise formulation makes temporal instruction following ambiguous. A single global prompt does not specify which sub-event should be realized in each chunk, while naively switching to step-wise prompts often leads to delayed reactions, blended step semantics, and error propagation across prompt transitions. These failures are difficult to address with supervised fine-tuning (SFT) or distillation alone: SFT suffers from exposure bias, while rollout-based distillation still optimizes low-level denoising or teacher-distribution matching rather than directly enforcing action ordering and prompt-transition correctness. We propose **TempAct**, **a planner–executor reinforcement learning** framework that jointly optimizes temporal decomposition and step-conditioned execution for temporally plausible AR video generation. TempAct uses an LLM planner to explore span-aware step prompts that are executable by the video model, and trains an AR diffusion executor to follow these prompts under its own generated histories. Its key mechanism is hierarchical group exploration: candidate plans form planning groups, and each plan induces an execution group of multiple continuations from a shared visual context, enabling plan-level credit assignment for long-horizon temporal outcomes and executor-level credit assignment for prompt-switch behavior. We further design hierarchical rewards that combine plan-quality and full-video temporal feedback for the planner with local transition-level step-following rewards, aesthetic reg-

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.

ularization, and KL constraints for the executor. Experiments on Self-Forcing and LongLive show that TempAct improves temporal consistency while preserving overall visual quality. Detailed comparisons of video generation results are provided in a **local HTML page** included in the supplementary materials.

1 INTRODUCTION

Video diffusion models (Kuaishou, 2024; Wan et al., 2025; Chen et al., 2025; Jin et al., 2025; Wang et al., 2025b; ByteDance, 2026; Wang et al., 2025a) have advanced rapidly in visual fidelity and motion realism, but deploying them in interactive applications requires more than high-quality offline generation: the model must generate streams with low latency while maintaining temporal coherence over long horizons. A common response to this demand is to distill pretrained video diffusion models into few-step autoregressive (AR) video generators (Huang et al., 2025; Cui et al., 2025; Yang et al., 2025b; Zhu et al., 2026), often using distribution-matching distillation (DMD) (Yin et al., 2024a) to preserve generation quality while reducing sampling cost. In this formulation, the model no longer synthesizes an entire video at once; instead, it generates one chunk after another, using previously generated frames as causal visual context. This context is stored through mechanisms such as attention states, KV caches, or visual caches, allowing subsequent chunks to be produced efficiently while preserving continuity with the past.

This chunk-wise formulation improves efficiency, but it can also make temporal instruction following ambiguous. In most AR generators, each chunk is typically conditioned on the same global prompt. As a result, the model knows the full instruction but not which part of the instruction should be realized in the current chunk. For example, in the dog example in Figure 1, the instruction requires the dog to first crouch toward the ball, then pick it up and return it to its owner. However, the single-prompt Self-Forcing baseline prematurely depicts the dog holding the ball while sitting at the beginning, causing an event from a later stage to appear too early. We refer to this failure as *temporal confusion*: the video may look reasonable at each frame, but **the event order is collapsed or blurred across chunks**. A straightforward fix is to replace the global prompt with step-wise prompts that specify the active subgoal for each stage. However, this shifts the difficulty to handling prompt transitions. As illustrated in Figures 1 and 3, with additional LongLive examples in **Appendix G**, when the prompt changes, the AR executor may keep following the previous action, blend adjacent step semantics, or carry errors from earlier chunks into later ones.

These failures are difficult to eliminate with supervised fine-tuning (SFT) or distillation alone. SFT suffers from exposure bias because it trains on clean teacher-forced histories but must infer from imperfect rollouts at test time (Schmidt, 2019; Ning et al., 2024). Rollout-based distillation methods such as DMD can reduce this mismatch, but both SFT and distillation still optimize low-level denoising or teacher-distribution matching (Lipman et al., 2022), rather than explicitly enforcing action ordering or prompt-transition correctness. This motivates reinforcement learning (Shao et al., 2024; Liu et al., 2025a): by exploring generated plans and videos and scoring them with LLM/VLM feedback (Guo et al., 2025; Jaech et al., 2024; Bai et al., 2025), RL can directly reward **executable temporal plans**, **successful prompt-switch execution**, and **long-horizon consistency**.

Motivated by this, we propose **TempAct**, a planner–executor reinforcement learning framework for temporally plausible AR video generation. Our key observation is that temporally complex generation requires solving two coupled problems: first determining *what* should happen at each stage, and then deciding *how* to realize that stage under autoregressive visual history. Executor-only training mainly improves the second problem, i.e., how well the video model follows a given condition, but it cannot revise an ambiguous or poorly timed temporal decomposition. A planner–executor framework therefore provides a more general solution: the planner can organize event order, step granularity, and state changes, while the executor learns to realize prompt transitions without drifting from the generated visual context. TempAct instantiates this idea by using an LLM planner to explore span-aware decompositions of a global instruction and an autoregressive video diffusion executor to roll out chunks conditioned on the active step and accumulated visual history. Crucially, **the LLM is not used as a fixed preprocessing module**; instead, TempAct optimizes both the planner and executor from generated trajectories, allowing the planner to learn which temporal plans are executable by the video model and the executor to learn how to realize prompt switches under its own imperfect histories.

To make this joint optimization effective, TempAct introduces a **hierarchical group exploration strategy** in which execution groups are nested under planning groups. For each global prompt, the planner samples a group of candidate temporal plans, covering different action orders, temporal granularities, and state descriptions. Conditioned on each candidate plan, the executor constructs a plan-specific execution group: it first generates a shared context from an earlier step, and then samples multiple continuations after the step transition under that same context. This nested structure provides aligned credit assignment at two levels: comparisons across planning groups reward decompositions that lead to better full-video temporal outcomes, while comparisons within each execution group isolate prompt-switch execution from variations in preceding visual history. Rewards follow the same hierarchy, combining plan-quality and full-video temporal-following feedback for the planner with local transition-level rewards and aesthetic regularization for the executor.

Empirically, we evaluate temporal plausibility using a Temporal-Following Score that measures whether generated videos follow the intended event sequence. TempAct improves this score across Self-Forcing and LongLive backbones on temporally complex video generation tasks, with consistent gains under both in-domain and out-of-domain VLM evaluation, as well as human evaluation.

2 Related Work

Autoregressive video generation. Causal distillation and rollout-based training enable efficient streaming video generation while mitigating exposure bias (Yin et al., 2024b; Huang et al., 2025; Cui et al., 2025; Zhu et al., 2026). Memory-management methods improve long-rollout consistency (Ji et al., 2025; 2026; Yi et al., 2025; Yang et al., 2026), while LongLive supports interactive generation through KV-recache, streaming long tuning, and frame-sink attention (Yang et al., 2025b). These approaches improve generation under a given text condition; TempAct instead jointly optimizes temporal decomposition and prompt-transition execution.

Reinforcement learning for generative models. Flow-GRPO and DanceGRPO extend group-relative optimization to visual generation through stochastic sampling (Liu et al., 2025a; Xue et al., 2025). Subsequent work improves sampling efficiency, credit assignment, clipping, and regularization (Li et al., 2025a;b; Wang et al., 2026b; He et al., 2025), while video feedback learning improves quality and alignment (Liu et al., 2025b). For AR video generation, AR-CoPO uses chunk-level forking and localized contrastive updates (He et al., 2026); Astrolabe uses forward-process fine-tuning and streaming local-window updates (Zhang et al., 2026b); and KVPO combines KV-cache semantic exploration with ODE-native optimization (Zhang et al., 2026a). These methods primarily align the executor. TempAct additionally learns how to organize complex instructions into executable temporal plans, coupling planner and executor updates through hierarchical credit assignment. **Appendix A** provides an expanded discussion. **Preliminaries.** Background on autoregressive video diffusion, flow matching, and group-relative policy optimization is provided in **Appendix B**.

3 Method

As illustrated in Figure 2, TempAct jointly optimizes an LLM planner π_ϕ for span-aware temporal decomposition and an autoregressive diffusion executor p_θ for step-conditioned generation under model-generated visual histories.

3.1 Hierarchical Planner-Executor Pipeline

LLM-based Step Planning and Prompt Enhancement. Given a global instruction $\boldsymbol{c}$ and a target latent video length K, we assign each step prompt a temporal span of L latent frames. The planner π_ϕ therefore produces a temporally ordered plan $\mathbf{s} = \{s_1, \ldots, s_J\}$ with $J = \lceil K/L \rceil$, where each step prompt s_j describes the action to be executed and the expected visual state during its assigned L-frame span. For example, generating 30 latent frames with $L = 6$ requires the LLM to decompose the prompt into $J = 5$ step prompts. Rather than using a deterministic decomposition, we sample a group of candidate plans from the LLM policy,

$$\mathbf{s}^m \sim \pi_\phi(\mathbf{s} \mid \boldsymbol{c}), \quad m = 1, \ldots, M, \tag{1}$$

where M is the planning-level group size. Each sampled plan contains both a step decomposition and prompt enhancement: the original temporal instruction is split into span-aware subgoals, and

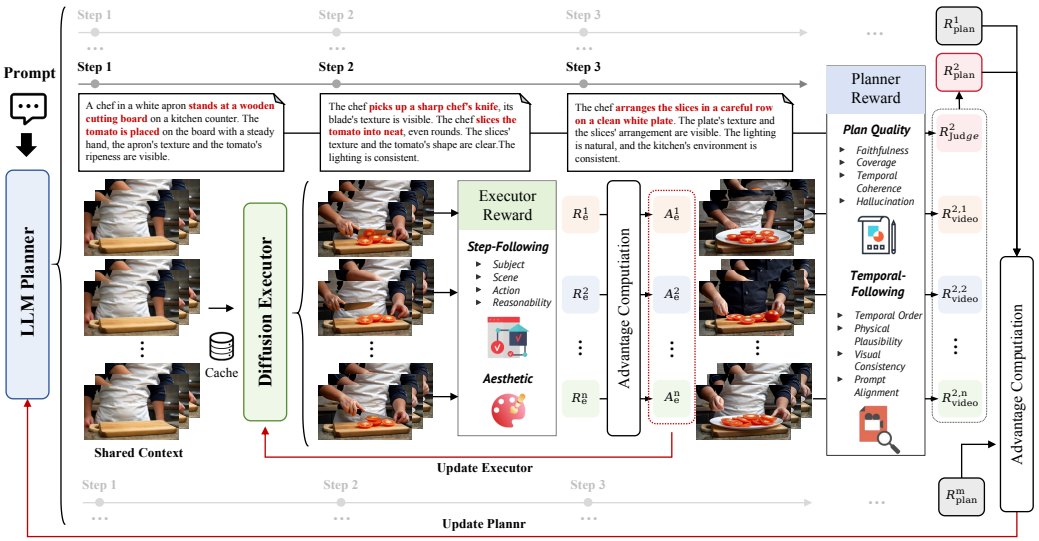


Figure 2: Overview of TempAct. An LLM planner samples span-aware temporal decompositions of a global instruction, while an autoregressive video executor rolls out shared contexts and multiple continuations under the corresponding step prompts. The nested planning–execution groups support hierarchical credit assignment, with plan-level rewards guiding executable decompositions and local transition-level rewards improving prompt-switch execution.

each subgoal is rewritten with explicit objects, actions, and state constraints for its assigned latent-frame interval. This stochastic planning stage allows the LLM to explore alternative temporal decompositions and receive reward feedback on which plans are more executable for the video generator. We further apply step-prompt smoothing for execution: instead of conditioning the diffusion model only on the current step, each span is represented by a smoothed prompt $\tilde{s}_j = \mathrm{Smooth}(s_j, s_{j+1})$ that exposes the executor to both the current subgoal and the upcoming subgoal. This transition-aware prompt reduces abrupt semantic changes between adjacent spans and empirically prevents large performance drops caused by hard prompt switching.

Hierarchical Autoregressive Video Sampling. For each sampled plan $\mathbf{s}^m$, the executor p_θ generates K latent video frames autoregressively under the smoothed step prompt assigned to each temporal span, where K is the target latent length defined above. To train the executor at a step transition from s_j^m to s_{j+1}^m, we first generate a shared context $\mathbf{x}_{1:k}^m$ up to the end of the current span, where $k = \min(jL, K)$. Each frame i is conditioned on the smoothed prompt of its own span $\sigma(i) = \lceil i/L \rceil$, where $\tilde{s}_{\sigma(i)}^m = \mathrm{Smooth}(s_{\sigma(i)}^m, s_{\sigma(i)+1}^m)$,

$$\mathbf{x}_{1:k}^m \sim \prod_{i=1}^{k} p_\theta(\mathbf{x}_i \mid \mathbf{x}_{<i}, \tilde{s}_{\sigma(i)}^m), \tag{2}$$

then sample N possible continuations $\{\mathbf{y}_{k+1:K}^{m,n}\}_{n=1}^N$ that complete the remaining video from span $j+1$ to the end. For each future latent frame, the executor conditions on the shared context, previously generated continuation frames, and the smoothed prompt assigned to that frame's temporal span,

$$\mathbf{y}_{k+1:K}^{m,n} \sim \prod_{i=k+1}^{K} p_\theta(\mathbf{y}_i^{m,n} \mid \mathbf{x}_{1:k}^m, \mathbf{y}_{k+1:i-1}^{m,n}, \tilde{s}_{\sigma(i)}^m), \quad n = 1, \ldots, N. \tag{3}$$

This yields a hierarchical group $\{\mathbf{s}^m, \mathbf{x}_{1:k}^m, \{\mathbf{y}_{k+1:K}^{m,n}\}_{n=1}^N\}_{m=1}^M$: the outer group compares different span-aware LLM plans, while the inner group compares different full-video continuations starting from the same context. The full continuations are used for reward evaluation, but the executor update is applied only to the first chunk after the prompt switch, so the credit assignment focuses on prompt-switch execution rather than all future spans.

3.2 JOINT RL OF PLANNER AND EXECUTOR

The sampled trajectories are scored by rewards defined in the next subsection. Let R^m_{plan} denote the planner-level reward for plan $\mathbf{s}^m$, computed from plan executability and video-level temporal instruction following. Let $R^{m,n}_{\text{exec}}$ denote the executor-level reward computed only on the first chunk after the prompt switch, measuring local step following and visual quality at the prompt-switch moment. We compute group-relative advantages separately at the two levels,

$$A^m_{\text{plan}} = \frac{R^m_{\text{plan}} - \mu_{\text{plan}}}{\sigma_{\text{plan}} + \epsilon}, \quad A^{m,n}_{\text{exec}} = \frac{R^{m,n}_{\text{exec}} - \mu^m_{\text{exec}}}{\sigma^m_{\text{exec}} + \epsilon}, \tag{4}$$

where μ_{plan} and σ_{plan} are computed over the M sampled plans, while μ^m_{exec} and σ^m_{exec} are computed over the N transition segments sharing the same plan and context. The LLM planner is updated with Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025). Following GSPO, we define the sequence importance ratio for a sampled plan as

$$\rho^m_\phi = \left(\frac{\pi_\phi(\mathbf{s}^m \mid \boldsymbol{c})}{\pi_{\phi_{\text{old}}}(\mathbf{s}^m \mid \boldsymbol{c})} \right)^{1/|\mathbf{s}^m|}, \tag{5}$$

where $|\mathbf{s}^m|$ denotes the number of generated planner tokens. The planner objective is

$$\mathcal{L}_{\text{planner}}(\phi) = -\mathbb{E}\left[\frac{1}{M} \sum_{m=1}^{M} \min\left(\rho^m_\phi A^m_{\text{plan}}, \text{clip}(\rho^m_\phi, 1-\epsilon_{\text{p}}, 1+\epsilon_{\text{p}}) A^m_{\text{plan}} \right) \right], \tag{6}$$

We regularize the planner with $\beta_{\text{p}} D_{\text{KL}}(\pi_\phi \| \pi_{\text{ref}})$. Unlike token-level PPO ratios, this sequence-level ratio assigns one relative update weight to the entire LLM plan, matching the fact that rewards are obtained after executing the complete temporal decomposition.

The diffusion executor is optimized with a clipped Flow-GRPO objective on the first chunk after the transition to span $j+1$, $\mathbf{y}^{m,n}_{k+1:k+t}$, where t is the chunk size, rather than on the entire remaining continuation. This choice is important because group-relative policy gradients compare samples under the same conditioning state: within this transition chunk, all candidates share the same preceding context and newly activated step prompt, whereas later chunks are generated from sample-specific histories and therefore correspond to different policy inputs. It also improves credit assignment by applying the RL update exactly at the prompt-switch moment, where the action for the new subgoal is decided and any execution error can propagate autoregressively to subsequent chunks. The executor objective is:

$$\mathcal{L}_{\text{exec}}(\theta) = -\mathbb{E}_{m, \{\mathbf{y}^{m,n}_{k+1:k+t}\}_{n=1}^{N}} \left[\frac{1}{N} \sum_{n=1}^{N} \min\left(\hat{\rho}^{m,n}_\theta A^{m,n}_{\text{exec}}, \text{clip}(\hat{\rho}^{m,n}_\theta, 1-\epsilon_{\text{e}}, 1+\epsilon_{\text{e}}) A^{m,n}_{\text{exec}} \right) \right]. \tag{7}$$

Here $\rho^{m,n}_\theta$ is the likelihood ratio of the denoising trajectory for $\mathbf{y}^{m,n}_{k+1:k+t}$, and $\hat{\rho}^{m,n}_\theta = \text{RatioNorm}(\rho^{m,n}_\theta)$ is the normalized ratio from GRPO-Guard (Wang et al., 2026b). In practice, we also add the KL regularizer $\beta_{\text{e}} D_{\text{KL}}(p_\theta(\mathbf{y}_{k+1:k+t} \mid \mathbf{x}_{1:k}, \tilde{s}_{j+1}) \| p_{\text{ref}}(\mathbf{y}_{k+1:k+t} \mid \mathbf{x}_{1:k}, \tilde{s}_{j+1}))$ to constrain the transition-chunk policy from drifting too far from the reference diffusion policy. Together, these updates encourage the planner to discover step prompts that are temporally coherent and executable, and the executor to realize the active subgoal at the prompt-switch moment under imperfect autoregressive context.

3.3 MULTI-LEVEL REWARD DESIGN

Planner Rewards. The planner is rewarded from two complementary perspectives: whether the generated decomposition is a high-quality temporal plan before execution, and whether executing this plan leads to videos that follow the original temporal instruction.

- *Plan Quality Score.* We use Qwen3-8B (Yang et al., 2025a) as an LLM-based plan judge to evaluate each sampled plan $\mathbf{s}^m$ against the original prompt $\boldsymbol{c}$. The judge focuses on faithfulness to the original instruction, coverage of all requested events, temporal coherence across steps, and hallucination avoidance. This produces a plan-quality reward R^m_{judge} that encourages the planner to preserve the original event structure, distribute actions across meaningful visual stages, and avoid unsupported additions.

- *Temporal-Following Score.* Conventional semantic alignment metrics, such as CLIP score, ImageReward, and VideoReward (Liu et al., 2025b), are often insensitive to event ordering and long-range temporal dependencies. We therefore use a VLM-based evaluator to score each executed video. Given sampled video frames and the original prompt, the evaluator not only predicts a temporal-order score, but also assesses physical consistency, visual consistency, and text-video consistency at both frame-level and video-level granularity. These auxiliary criteria prevent the policy from optimizing temporal order alone while ignoring visual plausibility or prompt faithfulness. For each sampled plan $\mathbf{s}^m$, we average the video-level scores over its N execution continuations to obtain $R^m_{\text{video}} = \frac{1}{N}\sum_{n=1}^{N} R^{m,n}_{\text{video}}$. Due to the high cost of large-scale training-time evaluation with stronger proprietary VLMs (like Gemini), we use Qwen3-VL-8B as the training-time video evaluator.

The final planner-level reward combines the textual plan-quality signal and the realized video-level temporal-following signal as R^m_{plan}, balancing decomposition quality with executable temporal consistency. Detailed instructions, scoring rules, aggregation weights, and scoring examples are provided in **Appendix E**.

Executor Rewards. The executor is rewarded from two complementary perspectives: whether it correctly realizes the newly activated step prompt after a prompt switch, and whether this semantic optimization preserves visual quality. Both signals are computed only on the first transition span $\mathbf{y}^{m,n}_{k+1:k+L}$ so that the reward remains aligned with the transition-span Flow-GRPO update.

- *Local Step-Following Score.* Similar to the temporal-following score, we use a VLM-based evaluator to assess step-following performance. For each first transition span $\mathbf{y}^{m,n}_{k+1:k+L}$, we use Qwen3-VL-8B to compute a local step-following reward $R^{m,n}_{\text{step}}$ from its sampled frames and the localized step prompt s^m_{j+1}. The evaluator relies only on visible evidence and checks whether the required subjects, scene, objects, motion, and visual dynamics are present and temporally plausible. This reward provides direct feedback on prompt-switch execution, rather than assigning credit from the full remaining continuation.
- *Aesthetic Quality Score.* To prevent semantic optimization from degrading visual quality, we include an aesthetic reward $R^{m,n}_{\text{aes}}$ computed by PickScore (Kirstain et al., 2023) on the same transition chunk.

The final executor-level reward $R^{m,n}_{\text{exec}}$ combines the local step-following signal and the aesthetic-quality signal, encouraging the executor to realize the active subgoal while maintaining visually pleasing generations. Detailed instructions and formulas are provided in **Appendix E**.

4 Experiments

4.1 Experimental Setup

Implementation Details. We evaluate TempAct on Self-Forcing and LongLive using 5,000 temporally structured instructions generated by GPT-5.2, with a Qwen3-1.7B planner. Hierarchical sampling uses $M = 4$ candidate plans and $N = 8$ executor rollouts per plan. Training videos contain 117 frames (30 latent frames), with $L = 6$ latent frames per step; evaluation uses $L = 12$. Unless otherwise specified, ablations use Self-Forcing on the Temporal Order benchmark, with Temporal-Following Scores assigned by the in-domain Qwen3-VL judge. LoRA configurations, hardware details, and training budgets are provided in **Appendix D**.

Temporal Order Benchmark. Our benchmark tests event ordering and state transitions on two subsets: a Simple Set of 100 prompts with one to two ordered steps, and a Hard Set of 100 prompts with three to four steps and more complex dependencies. Evaluation fixes planner outputs to three and five steps, respectively, each spanning 12 latent frames.

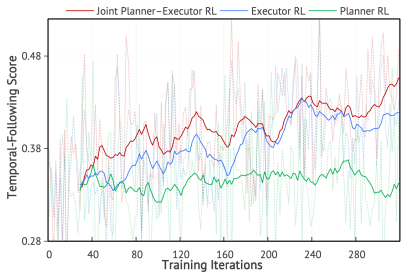
Evaluation Metrics. We evaluate general quality and semantic alignment with VBench (Huang et al., 2024) on 141-frame videos, aesthetic preference with PickScore on Temporal Order benchmark videos, and efficiency with throughput. Temporal-Following Scores use uniformly sampled full-video frames and the scoring protocol of $R^{m,n}_{\text{video}}$ (**Appendix E**). We report scores on both benchmark subsets using Qwen3-VL-8B, from the same model family as the training-time video reward model, and

Table 1: Comparison of different methods on the Temporal Order benchmark using Temporal-Following Score as the primary metric. Scores are in $[0, 1]$ and are evaluated on full-video frame samples; Avg. averages the four Temporal-Following Scores across the Simple and Hard sets.

Method	Temporal-Following Score					VBench			PickScore
	Simple Set (1-2 Steps)		Hard Set (3-4 Steps)		Avg.	Total	Quality	Semantic	
	Qwen	Gemini	Qwen	Gemini					
Single-prompt video generation									
Self-Forcing	0.410	0.456	0.240	0.419	0.381	81.20	83.80	70.50	20.7
Causal Forcing	0.381	0.447	0.304	0.452	0.396	80.85	84.02	68.18	20.8
LongLive	0.428	0.505	0.302	0.463	0.424	80.36	82.72	70.91	21.1
Step-prompt video generation									
Self-Forcing	0.414	0.485	0.269	0.431	0.400	80.07	82.89	68.82	20.6
+ TempAct	**0.500**$_{+20.8\%}$	**0.538**$_{+10.9\%}$	**0.336**$_{+24.9\%}$	**0.473**$_{+9.7\%}$	**0.462**$_{+15.5\%}$	79.99	82.71	69.14	20.6
LongLive	0.411	0.521	0.314	0.481	0.432	79.55	82.10	69.35	20.8
+ TempAct	**0.508**$_{+23.6\%}$	**0.579**$_{+11.1\%}$	**0.352**$_{+12.1\%}$	**0.512**$_{+6.4\%}$	**0.488**$_{+13.0\%}$	79.97	82.61	69.40	20.8

Table 2: Component ablation. **Left:** Temporal-Following Scores when progressively adding step-wise prompting, prompt smoothing, planner RL, and executor RL. **Right:** Training reward trajectories in the planner–executor RL ablation study. PE denotes prompt enhancement.

Setting	Simple Set↑	Hard Set↑	Avg.↑
Single Prompt	0.410	0.240	0.325
Single Prompt + PE RL	0.409	0.243	0.326
Step Prompt	0.287	0.208	0.248
Step Prompt + Smoothing	0.414	0.269	0.342
+ Planner RL	0.420	0.304	0.362
+ Executor RL	0.458	0.321	0.390
+ Joint Planner–Executor RL	0.500	0.336	0.418



Gemini 3 Flash, which is never used during training. These judges assess in-domain performance and out-of-domain generalization, respectively.

4.2 Quantitative and Qualitative Results

Quantitative analysis. Table 1 shows that TempAct improves temporal instruction following on both autoregressive backbones. Relative to step-prompt baselines, the average Temporal-Following Score rises from 0.400 to 0.462 on Self-Forcing (+15.5%) and from 0.432 to 0.488 on LongLive (+13.0%). Improvements extend to the Hard Set, where Qwen3-VL scores increase by 24.9% and 12.1%, respectively, supporting the effectiveness of joint planning and execution for multi-stage instructions. Gains are consistent across both benchmark subsets under the in-domain Qwen3-VL and out-of-domain Gemini 3 Flash judges, suggesting that the benefits generalize beyond the training-time reward model. Importantly, these temporal gains preserve overall visual quality. Self-Forcing's VBench Total changes only marginally (80.07 to 79.99), while LongLive improves across total, quality, and semantic scores. Both backbones achieve higher semantic scores, and PickScore remains unchanged at 20.6 and 20.8, respectively. Together, these results support improved event ordering and prompt-transition execution without a substantial trade-off in visual fidelity or aesthetic preference.

Qualitative analysis. Figure 3 compares single-prompt generation, step-prompt generation, and TempAct on Self-Forcing; LongLive examples appear in **Appendix G** (Figure 9). Single-prompt generation blends actions from different stages or omits requested events. Step prompts clarify the active subgoal but do not reliably resolve prompt-transition failures. In the squirrel example, the baseline leaves the acorn on the ground instead of burying it, failing to realize the requested state transition. TempAct produces more faithful event progression and action completion, illustrating how jointly optimizing temporal decomposition and execution reduces semantic confusion and improves adherence to ordered instructions.



Figure 3: Qualitative comparison on temporally ordered prompts using the Self-Forcing backbone.

4.3 Ablation Study

Core Component Ablation. Table 2 shows that prompt-enhancement RL alone offers negligible gains under single-prompt conditioning. Step prompts without smoothing degrade performance, while smoothing restores the average Temporal-Following Score from 0.248 to 0.342, highlighting the importance of stable prompt transitions. Building on this setup, planner RL and executor RL independently improve performance, with larger gains from executor RL. Joint optimization achieves the highest average score of 0.418, outperforming either component alone and demonstrating the complementary roles of executable temporal planning and step-conditioned execution. Additional ablations show that cross-plan advantage normalization and independent-prefix sampling reduce the average Temporal-Following Score from 0.418 to 0.374 and 0.369, respectively, supporting the contribution of the hierarchical design beyond joint optimization alone (**Appendix F**).

KL Regularization. Table 3 shows that $\beta_e = 5 \times 10^{-4}$ achieves the best average Temporal-Following Score (0.390) and PickScore (20.6). Removing KL reduces both metrics, while a smaller weight improves aesthetics without improving temporal following. A larger weight lowers temporal performance, suggesting that excessive regularization restricts useful updates.

Reward Ablation for Planner. Table 4 evaluates reward choices for planner optimization, where TA denotes the text-alignment metric from Liu et al. (2025b). Using the video-level reward R^m_{video} improves performance on both the Simple and Hard sets over the baseline, showing that execution feedback can guide the planner toward more useful decompositions. Adding the textual judge

Table 3: Ablation of executor KL regularization.

β_e	Simple Set↑	Hard Set↑	Avg.↑	PickScore↑
0	0.452	0.297	0.375	20.2
1×10^{-4}	0.439	0.309	0.374	20.4
5×10^{-4}	0.458	0.321	0.390	20.6
1×10^{-3}	0.452	0.313	0.383	20.5

Table 4: Ablation of planner reward design.

Planner Reward	Simple Set↑	Hard Set↑	TA↑
Baseline	0.414	0.269	-0.03
R^m_{video}	0.428	0.313	-0.06
$R^m_{\text{video}} + R^m_{\text{judge}}$	0.420	0.304	-0.04

reward R^m_{judge} slightly lowers the Temporal-Following Scores but improves TA from -0.06 to -0.04, indicating that the judge helps regularize plan quality while the video reward remains the main driver of temporal-following performance.

Reward Ablation for Executor. Table 5 shows that local step-following reward is the key signal for executor training. Using $R^{m,n}_{\text{step}}$ improves the average score from 0.342 to 0.390, while adding PickScore preserves the same Temporal-Following Score and recovers aesthetic quality to 20.6. In contrast, using only the full-video reward $R^{m,n}_{\text{video}}$ performs worse than no executor RL, and combining it with the local reward also reduces performance. This suggests that executor learning benefits most from rewards aligned with the prompt-switch transition segment, whereas full-video rewards are too coarse for low-level execution updates.

Table 5: Ablation of executor reward design.

Executor Reward	Simple Set↑	Hard Set↑	Avg.↑	PickScore↑
No Executor RL	0.414	0.269	0.342	20.6
$R^{m,n}_{\text{step}}$	0.465	0.314	0.390	20.4
$R^{m,n}_{\text{step}} + R^{m,n}_{\text{aes}}$	0.458	0.321	0.390	20.6
$R^{m,n}_{\text{step}} + R^{m,n}_{\text{video}}$	0.421	0.304	0.363	20.6
$R^{m,n}_{\text{video}}$	0.381	0.244	0.313	20.7

4.4 User Study

Human preference evaluation. To further validate whether the Temporal-Following gains reflect human judgments, we conduct a pairwise user study on the Temporal Order benchmark. For each prompt, annotators compare videos generated by TempAct and the corresponding baseline, and choose which video better preserves the intended temporal order and action progression, with ties allowed when the two videos are comparable. We collect preference statistics over 100 evaluated samples for each backbone and report the win/tie/loss ratios for Self-Forcing and LongLive. As shown in Figure 4, TempAct is preferred substantially more often than the baseline on both backbones, indicating that planner–executor RL improves temporal consistency in a way that is perceptible to human evaluators.

Agreement between Temporal-Following Scores and human preference. We also examine whether the proposed VLM-based Temporal-Following Score is aligned with human preference. For each TempAct–baseline pair in the human study, we compare the video preferred by annotators with the video assigned the higher Temporal-Following Score by the VLM judge. To account for ambiguous cases, we treat VLM score differences within ± 0.03 as ties. Under this protocol, Gemini 3 Flash agrees with human preference on 81% of the evaluated pairs, while Qwen3-VL-8B reaches 76% agreement. This high consistency suggests that Temporal-Following Score is a reliable proxy for human-perceived temporal plausibility, and that the improvements reported in Table 1 correspond to meaningful gains in user preference.

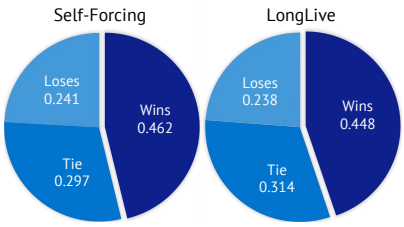


Figure 4: Human preference.

5 Conclusion

We presented TempAct, a planner–executor reinforcement learning framework that couples temporal decomposition with step-conditioned autoregressive video generation under model-generated visual histories. On Self-Forcing and LongLive, TempAct improves temporal instruction following under both in-domain and out-of-domain VLM evaluation and human assessment while preserving visual quality. Ablations demonstrate the complementary roles of planner and executor optimization and the importance of local transition-level rewards. These findings support joint planning and execution for temporally plausible streaming video generation. Limitations and future directions are discussed in **Appendix C**.

AI USE STATEMENT

We used generative AI tools to help refine the writing of this manuscript. We have reviewed all AI-assisted work and take responsibility for the final content of this research, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

ByteDance. Seedance. `https://seed.bytedance.com/zh/seedance2_0`, 2026.

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.

Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-Forcing++: Towards minute-scale high-quality video generation. *arXiv preprint arXiv:2510.02283*, 2025.

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*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dailan He, Guanlin Feng, Xingtong Ge, Yi Zhang, Bingqi Ma, Guanglu Song, Yu Liu, and Hongsheng Li. Ar-copo: Align autoregressive video generation with contrastive policy optimization. *arXiv preprint arXiv:2603.17461*, 2026.

Haoran He, Yuxiao Ye, Jie Liu, Jiajun Liang, Zhiyong Wang, Ziyang Yuan, Xintao Wang, Hangyu Mao, Pengfei Wan, and Ling Pan. Gardo: Reinforcing diffusion models without reward hacking. *arXiv preprint arXiv:2512.24138*, 2025.

Jinyi Hu, Shengding Hu, Yuxuan Song, Yufei Huang, Mingxuan Wang, Hao Zhou, Zhiyuan Liu, Wei-Ying Ma, and Maosong Sun. Acdit: Interpolating autoregressive conditional modeling and diffusion transformer. *arXiv preprint arXiv:2412.07720*, 2024.

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, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Sihui Ji, Xi Chen, Shuai Yang, Xin Tao, Pengfei Wan, and Hengshuang Zhao. Memflow: Flowing adaptive memory for consistent and efficient long video narratives. *arXiv preprint arXiv:2512.14699*, 2025.

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.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations*, volume 2025, pp. 23378–23402, 2025.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-Pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663, 2023.

Kuaishou. Kling. `https://klingai.kuaishou.com/`, 2024.

Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Yiming Cheng, Miles Yang, Zhao Zhong, and Liefeng Bo. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE-SDE. *arXiv preprint arXiv:2507.21802*, 2025a.

Yuming Li, Yikai Wang, Yuying Zhu, Zhongyu Zhao, Ming Lu, Qi She, and Shanghang Zhang. BranchGRPO: Stable and efficient GRPO with structured branching in diffusion models. *arXiv preprint arXiv:2509.06040*, 2025b.

Yuming Li, Qingyu Li, Chengyu Bai, Xiangyang Luo, Zeyue Xue, Wenyu Qin, Meng Wang, Yikai Wang, and Shanghang Zhang. Aegpo: Adaptive entropy-guided policy optimization for diffusion models. *arXiv preprint arXiv:2602.06825*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025a.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, et al. Improving video generation with human feedback. *Advances in Neural Information Processing Systems*, 38:82155–82192, 2025b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 15167–15189, 2024.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pp. 8857–8868. PMLR, 2021.

Florian Schmidt. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 157–167, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Jin Wang, Jianxiang Lu, Guangzheng Xu, Comi Chen, Haoyu Yang, Linqing Wang, Peng Chen, Mingtao Chen, Zhichao Hu, Longhuang Wu, et al. Tagrpo: Boosting grpo on image-to-video generation with direct trajectory alignment. *arXiv preprint arXiv:2601.05729*, 2026a.

Jing Wang, Ao Ma, Ke Cao, Jun Zheng, Jiasong Feng, Zhanjie Zhang, Wanyuan Pang, and Xiaodan Liang. WISA: World simulator assistant for physics-aware text-to-video generation. *Advances in Neural Information Processing Systems*, 38:5388–5416, 2025a.

Jing Wang, Ao Ma, Jiasong Feng, Dawei Leng, Yuhui Yin, and Xiaodan Liang. Pt-t2i/v: An efficient proxy-tokenized diffusion transformer for text-to-image/video-task. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Jing Wang, Jiajun Liang, Jie Liu, Henglin Liu, Gongye Liu, Jun Zheng, Wanyuan Pang, Ao Ma, Zhenyu Xie, Xintao Wang, et al. GRPO-Guard: Mitigating implicit over-optimization in flow matching via regulated clipping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5988–5998, 2026b.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

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*, 2025b.

Yang Yang, Tianyi Zhang, Wei Huang, Jinwei Chen, Boxi Wu, Xiaofei He, Deng Cai, Bo Li, and Peng-Tao Jiang. Anchor forcing: Anchor memory and tri-region rope for interactive streaming video diffusion. *arXiv preprint arXiv:2603.13405*, 2026.

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, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024a.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast causal video generators. *arXiv preprint arXiv:2412.07772*, 2024b.

Ruicheng Zhang, Kaixi Cong, Jun Zhou, Zhizhou Zhong, Zunnan Xu, Shuiyang Mao, Wei Liu, and Xiu Li. KVPO: ODE-native GRPO for autoregressive video alignment via KV semantic exploration. *arXiv preprint arXiv:2605.14278*, 2026a.

Songchun Zhang, Zeyue Xue, Siming Fu, Jie Huang, Xianghao Kong, Y. Ma, Haoyang Huang, Nan Duan, and Anyi Rao. Astrolabe: Steering forward-process reinforcement learning for distilled autoregressive video models. *arXiv preprint arXiv:2603.17051*, 2026b.

Min Zhao, Hongzhou Zhu, Kaiwen Zheng, Zihan Zhou, Bokai Yan, Xinyuan Li, Xiao Yang, Chongxuan Li, and Jun Zhu. Causal forcing++: Scalable few-step autoregressive diffusion distillation for real-time interactive video generation. *arXiv preprint arXiv:2605.15141*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 2026.

A Extended Related Work

A.1 Autoregressive Video Generation

Autoregressive (AR) video generation enables streaming synthesis by producing frames or chunks sequentially, but teacher-forced AR diffusion models suffer from train-test mismatch and error accumulation when conditioned on their own histories (Hu et al., 2024; Gao et al., 2024). Recent work improves AR generation through causal distillation, rollout-based training, and memory management: CausVid converts bidirectional diffusion transformers into causal generators with video DMD and KV caching (Yin et al., 2024b;a); Self-Forcing trains with autoregressive rollouts and video-level feedback to mitigate exposure bias (Huang et al., 2025; Cui et al., 2025); subsequent methods manage contextual memory for more consistent long rollouts (Ji et al., 2025; 2026; Yi et al., 2025; Yang et al., 2026); LongLive further introduces KV-recache, streaming long tuning, and frame-sink attention for interactive long-video generation (Yang et al., 2025b); and Causal Forcing bridges the architectural gap between bidirectional teachers and causal students by using an AR teacher for ODE initialization before DMD post-training (Zhu et al., 2026; Zhao et al., 2026). These methods improve streaming efficiency and rollout stability under a given text condition, whereas TempAct focuses on instruction-guided temporal plausibility by jointly optimizing temporal planning and prompt-transition execution with hierarchical reinforcement learning.

A.2 Reinforcement Learning for Generative Models

The success of group-relative policy optimization in language models has inspired growing interest in reinforcement learning for generative models. For diffusion and flow-matching models, Flow-GRPO and DanceGRPO introduce stochastic sampling paths that make GRPO-style policy updates applicable to visual generation (Liu et al., 2025a; Xue et al., 2025). Subsequent studies improve the efficiency and stability of online diffusion RL through better sampling schemes, credit assignment, clipping strategies, and regularization (Li et al., 2025a; Wang & Yu, 2025; Li et al., 2025b; 2026; Wang et al., 2026b; He et al., 2025; Wang et al., 2026a). Beyond these optimizer and sampler designs, video feedback learning shows that preference and semantic rewards can improve video generation quality and alignment (Liu et al., 2025b).

Recent work further applies RL to autoregressive video generation. AR-CoPO (He et al., 2026) uses chunk-level forking and localized contrastive updates. Astrolabe (Zhang et al., 2026b) aligns distilled AR models with forward-process negative-aware fine-tuning and streaming local-window updates. KVPO (Zhang et al., 2026a) introduces ODE-native GRPO with KV-cache semantic exploration and velocity-space contrastive objectives. Together, these methods advance executor-level preference alignment for streaming AR generators, but they do not explicitly model how complex instructions should be temporally organized, leaving event ordering and long-horizon semantic dependencies under-specified. In contrast, TempAct performs planner–executor RL over temporally complex instructions, using an LLM planner to explore executable temporal decompositions and an AR executor to learn prompt-switch execution under hierarchical credit assignment, offering a new perspective on long-horizon temporal plausibility.

B Additional Background

Autoregressive Video Diffusion Models. Autoregressive (AR) video diffusion models provide the basic streaming formulation used by recent long-video generation methods such as Self-Forcing (Huang et al., 2025) and LongLive (Yang et al., 2025b). Given a text condition $\boldsymbol{c}$, an AR video model represents a video sequence $\mathbf{x}_{1:K} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ through the causal factorization

$$p_\theta(\mathbf{x}_{1:K} \mid \boldsymbol{c}) = \prod_{i=1}^{K} p_\theta(\mathbf{x}_i \mid \mathbf{x}_{<i}, \boldsymbol{c}), \tag{8}$$

where $\mathbf{x}_i$ denotes the i-th generated frame or video chunk, and $\mathbf{x}_{<i} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}\}$ denotes the previously generated frames or chunks used as causal visual context. Generation then proceeds sequentially, sampling each $\mathbf{x}_i$ conditioned on $\mathbf{x}_{<i}$ and $\boldsymbol{c}$.

Each conditional distribution $p_\theta(\mathbf{x}_i \mid \mathbf{x}_{<i}, \boldsymbol{c})$ is modeled by flow matching (Lipman et al., 2022; Liu et al., 2022). To train the model, for the i-th frame or chunk $\mathbf{x}_i$, a noise endpoint $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and

a timestep $t \sim \mathcal{U}(0,1)$ are sampled, and the intermediate state is defined as $\mathbf{x}_i^t = (1-t)\mathbf{x}_i + t\boldsymbol{\epsilon}_i$. The model predicts the velocity field $\boldsymbol{v}_\theta(\mathbf{x}_i^t, t \mid \mathbf{x}_{<i}, \boldsymbol{c})$ from the noisy state, timestep, text condition, and causal visual context. The final training loss is the conditional flow-matching (CFM) objective, defined as

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{i,t,\mathbf{x}_i,\boldsymbol{\epsilon}_i}\left[\left\|\boldsymbol{v}_\theta(\mathbf{x}_i^t, t \mid \mathbf{x}_{<i}, \boldsymbol{c}) - (\boldsymbol{\epsilon}_i - \mathbf{x}_i)\right\|_2^2\right]. \tag{9}$$

The main difference among AR training schemes lies in how the historical context is constructed. Teacher forcing (TF) (Williams & Zipser, 1989; Rasul et al., 2021) conditions each prediction on clean ground-truth history $\mathbf{x}_{<i}$. Diffusion forcing (DF) (Chen et al., 2024) improves robustness by corrupting the historical frames with independently sampled noise levels, so the model learns to denoise under noisy contexts $\{\mathbf{x}_j^{t_j}\}_{j<i}$. However, both TF and DF still build the history from data rather than from the model's own generated frames, causing exposure bias when the model must condition on its own outputs at inference time. Self-Forcing (Huang et al., 2025) addresses this mismatch by training on rollouts from the current model, $\mathbf{x}_{1:K}^\theta \sim \prod_{i=1}^K p_\theta(\mathbf{x}_i \mid \mathbf{x}_{<i}, \boldsymbol{c})$, and distilling teacher velocity predictions along these self-generated trajectories.

GRPO and Flow-GRPO. Group Relative Policy Optimization (GRPO) is a critic-free policy optimization algorithm for improving LLM reasoning ability (Shao et al., 2024). For a prompt q, the current policy π_θ samples a group of G responses $\{o_i\}_{i=1}^G$, each scored by a reward function $r_i = r(q, o_i)$. GRPO estimates the advantage of each response by normalizing rewards within the same group:

$$A_i = \frac{r_i - \mathrm{mean}(\{r_j\}_{j=1}^G)}{\mathrm{std}(\{r_j\}_{j=1}^G) + \epsilon}. \tag{10}$$

The policy is then updated with a PPO-style clipped objective and a reference-policy regularizer:

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\rho_i A_i, \mathrm{clip}(\rho_i, 1-\epsilon_c, 1+\epsilon_c)A_i\right) - \beta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})\right], \tag{11}$$

where $\rho_i = \pi_\theta(o_i \mid q)/\pi_{\theta_{\mathrm{old}}}(o_i \mid q)$ and π_{ref} is a fixed reference policy. The key property of GRPO is that relative comparison among samples from the same prompt provides a low-cost baseline, avoiding the extra critic model required by PPO (Schulman et al., 2017). Group Sequence Policy Optimization (GSPO) extends this group-relative principle to sequence-level policy updates by normalizing the importance ratio over the generated sequence, providing a stable update rule for long-form LLM generations (Zheng et al., 2025).

Flow-GRPO extends group-relative policy optimization to diffusion and flow-matching generators (Liu et al., 2025a). Since standard flow-matching samplers follow deterministic ODE trajectories, Flow-GRPO uses an ODE-to-SDE conversion to recast generation as a stochastic policy, making it compatible with GRPO-style optimization. In flow-matching RL, however, the importance-ratio distribution can be biased across denoising timesteps, which weakens the intended clipping constraint and may lead to over-optimization. GRPO-Guard (Wang et al., 2026b) addresses this issue with regulated clipping, including ratio normalization $\hat{\rho} = \mathrm{RatioNorm}(\rho)$ that recenters and stabilizes the timestep-wise ratio distribution before applying the clipped surrogate.

C Limitations and Future Work

Although TempAct effectively improves temporal plausibility, it is still constrained by the capability of the underlying autoregressive video generator. As a result, some failure cases may remain, including visual distortions, physically implausible motions, or accumulated artifacts in long rollouts. In addition, the current planner assigns step prompts to fixed temporal spans, which may be suboptimal for actions with different durations or varying execution difficulty. A promising direction is to make the planner duration-aware, allowing it to allocate generation time adaptively according to the semantic complexity of each action. Another future direction is to extend the LLM planner into a VLM-based planner that can observe previously generated content and dynamically adjust subsequent instructions based on the actual visual state. Such closed-loop planning could further improve long-horizon consistency and make autoregressive video generation more robust under complex temporal instructions.

D TRAINING HYPERPARAMETERS

Table 6 summarizes the main hyperparameters used for TempAct training. We separate the configurations into the shared autoregressive video setup, planner optimization, executor optimization, hierarchical reinforcement learning, and streaming rollout settings.

D.1 FULL EXPERIMENTAL SETUP

Implementation Details. We evaluate TempAct on two autoregressive video diffusion backbones, Self-Forcing and LongLive, using a training set of 5,000 temporally structured instructions generated by GPT-5.2. For each training instruction, TempAct first samples $M = 4$ candidate plans from the planner and then samples $N = 8$ executor rollouts for each plan, producing 32 videos for hierarchical credit assignment. Each sampled video contains 117 frames, which correspond to 30 latent frames. To match the training rollout length, each semantic step spans $L = 6$ latent frames during training, while evaluation uses a longer span of $L = 12$ latent frames per step. The planner is initialized from Qwen3-1.7B and optimized with LoRA using rank 32 and alpha 64. The diffusion executor is optimized separately with LoRA using rank 256 and alpha 256. Training uses 32 NVIDIA H20 GPUs with a batch size of 16; evaluation also uses NVIDIA H20 GPUs. Each training step takes approximately 800 seconds, and the models are trained for 300–400 steps. Detailed training hyperparameters are summarized in Table 6.

Temporal Order Benchmark. To evaluate temporal instruction following beyond generic video quality, we construct a Temporal Order benchmark that emphasizes whether generated videos preserve the requested event order and state transitions. The benchmark contains two subsets with different levels of temporal complexity: the Simple Set includes 100 prompts with one to two ordered steps, and the Hard Set includes 100 prompts with three to four ordered steps and more complex dependencies. For each generated video, we uniformly sample frames from the full video and ask VLM evaluators to judge whether the video follows the intended temporal order and realizes the described actions or state changes. During evaluation, we fix the planner output length to 3 steps for the Simple Set and 5 steps for the Hard Set. Each step covers 12 latent frames, i.e., $L = 12$.

Evaluation Metrics. We report both general video-generation metrics and temporal instruction-following metrics. For general quality and semantic alignment, we evaluate VBench on 141-frame generated videos. We also report PickScore as an aesthetic preference metric and throughput as an efficiency metric, with PickScore computed on videos generated for the Temporal Order benchmark. For temporal instruction following, we report Temporal-Following Scores on both the Simple and Hard Sets of the Temporal Order benchmark. We use two VLM judges for this evaluation: Qwen3-VL-8B, which belongs to the same model family as the training-time video reward model and serves as an in-domain evaluator, and Gemini 3 Flash, which is never used during training and serves as an out-of-domain evaluator for testing reward-model generalization. Both evaluators follow the same scoring protocol as $R^{m,n}_{\text{video}}$.

D.2 HYPERPARAMETER SUMMARY

E REWARD DETAILS

E.1 PLANNER REWARDS

Plan Quality Score. The LLM-based plan judge returns faithfulness F^m, coverage C^m, temporal coherence T^m, and hallucination H^m scores in a structured JSON format:

```
{
  "faithfulness_score": 0.0,
  "coverage_score": 0.0,
  "temporal_score": 0.0,
  "hallucination_score": 0.0
}
```

The hallucination score measures unsupported content introduced by the plan: 0.0–0.2 indicates minor harmless visual enrichment, 0.2–0.5 indicates noticeable unsupported descriptive additions,

and 0.5–1.0 indicates significant invented details, events, or scene changes. Larger hallucination therefore indicates more unsupported content. We aggregate these scores into

$$R^m_{\text{judge}} = 0.4F^m + 0.25C^m + 0.25T^m - 0.3H^m. \tag{12}$$

Temporal-Following Score. The video-level evaluator returns a structured JSON response that contains three parts: (i) a reasoning field with frame-level evidence, detected errors, and aggregation logic; (ii) frame-level scores for the sampled frames; and (iii) global scores for temporal order, physical plausibility, visual consistency, prompt alignment, and overall quality. A typical response follows the schema

```
{
  "think": {
    "frame_evidence": ["frame 1: ...", "frame 2: ..."],
    "detected_errors": ["..."],
    "aggregation_logic": "..."
  },
  "frame_scores": [0, 0, 0, 0, 0, 0],
  "temporal_order_score": 0,
  "physical_plausibility_score": 0,
  "visual_consistency_score": 0,
  "prompt_alignment_score": 0,
  "final_score": 0
}
```

where all scalar scores are returned on a 0–10 scale. We normalize each score to $[0, 1]$ and aggregate them as

$$\begin{aligned} R^{m,n}_{\text{video}} = 0.2\,\bar{S}^{m,n}_{\text{frame}} + 0.3\,S^{m,n}_{\text{final}} + 0.2\,S^{m,n}_{\text{temp}} \\ + 0.1\,S^{m,n}_{\text{phys}} + 0.1\,S^{m,n}_{\text{vis}} + 0.1\,S^{m,n}_{\text{align}}, \end{aligned} \tag{13}$$

where $\bar{S}^{m,n}_{\text{frame}}$ is the average normalized frame-level score, and $S^{m,n}_{\text{final}}$, $S^{m,n}_{\text{temp}}$, $S^{m,n}_{\text{phys}}$, $S^{m,n}_{\text{vis}}$, and $S^{m,n}_{\text{align}}$ denote the normalized final, temporal-order, physical-plausibility, visual-consistency, and prompt-alignment scores, respectively. This aggregation follows the same weighting scheme as our VLM reward implementation and prevents the planner from optimizing only temporal order while ignoring physical realism, visual stability, or text-video alignment.

The planner-level reward is

$$R^m_{\text{plan}} = 0.15R^m_{\text{judge}} + 0.85R^m_{\text{video}}. \tag{14}$$

E.2 Executor Rewards

Local Step-Following Score. The local step-following evaluator returns a structured JSON response with reasoning fields and scalar scores for the transition segment:

```
{
  "think": {
    "detected_errors": ["..."],
    "temporal_reasoning": ["..."],
    "motion_smooth_and_blur_reasoning": ["..."]
  },
  "subject_score": 0,
  "scene_score": 0,
  "action_score": 0,
  "object_appearance_reasonability_score": 0,
  "final_score": 0
}
```

The scalar scores measure subject correctness, scene consistency, action execution, object-appearance plausibility, and overall step-following quality, denoted by $S^{m,n}_{\text{sub}}$, $S^{m,n}_{\text{scene}}$, $S^{m,n}_{\text{act}}$, $S^{m,n}_{\text{obj}}$, and $S^{m,n}_{\text{final}}$. We aggregate them into

$$R^{m,n}_{\text{step}} = 0.15S^{m,n}_{\text{sub}} + 0.15S^{m,n}_{\text{scene}} + 0.2S^{m,n}_{\text{act}} + 0.2S^{m,n}_{\text{obj}} + 0.3S^{m,n}_{\text{final}}. \tag{15}$$

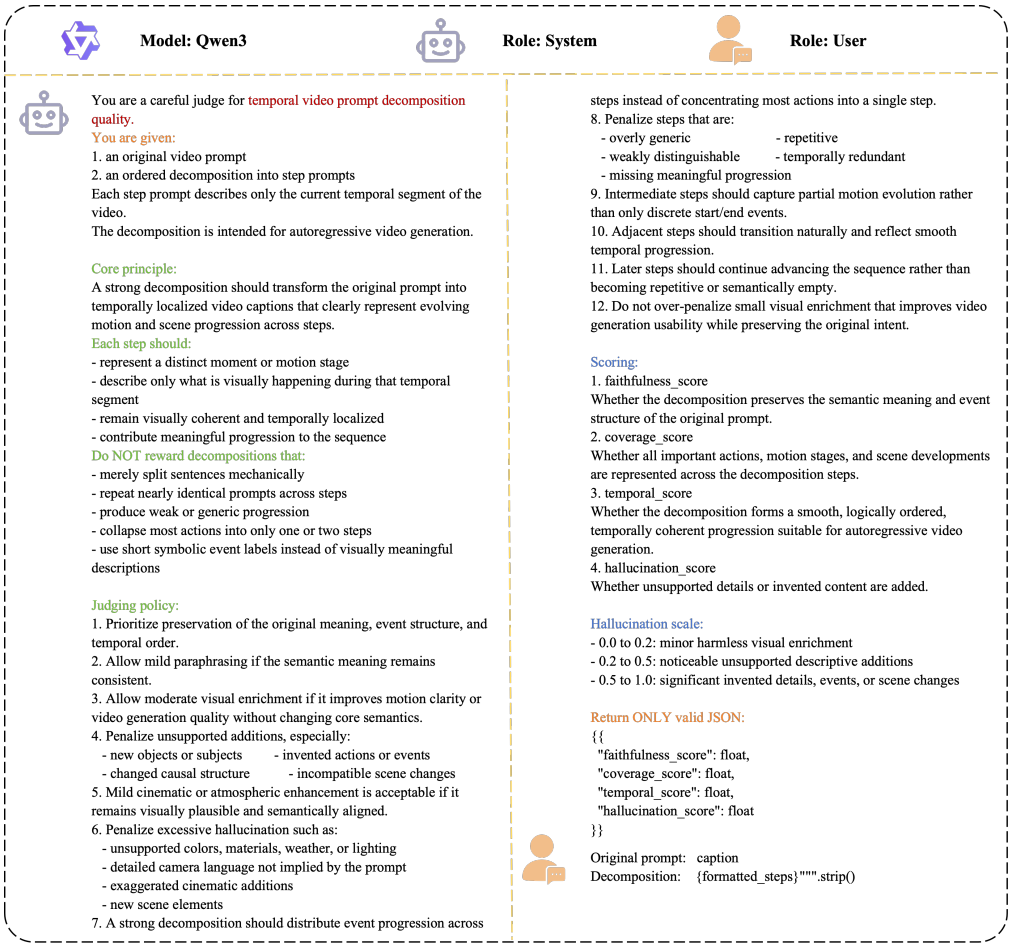


Figure 5: Prompt instructions for evaluating the Plan Quality Score.

Given the PickScore aesthetic reward $R_{\text{aes}}^{m,n}$ on the same transition segment, the executor-level reward is

$$R_{\text{exec}}^{m,n} = 0.6R_{\text{step}}^{m,n} + 0.4R_{\text{aes}}^{m,n}. \tag{16}$$

E.3 Prompt Instructions

We provide detailed LLM and VLM input instructions for evaluating the Plan Quality Score, Temporal-Following Score, and Local Step-Following Score, as shown in Figures 5, 6, and 7. We also provide examples of scoring results in Figure 8.

Model: Qwen3-VL / Gemini | **Role: System** | **Role: User**

You are a video consistency evaluator.
Your task is to determine how well a video matches a text prompt using STRICT evidence-based reasoning.

INPUT:
- A text prompt describing actions
- {num_sample_frames} frames in chronological order

CRITICAL PRINCIPLE:
You MUST base ALL scoring on explicit observable evidence from frames.
Do NOT:
- give scores based on overall impression
- guess missing errors without visual evidence
- assign uniformly high or low scores without justification

STEP 1: FRAME-BY-FRAME OBSERVATION
For EACH frame:
- describe visible objects and actions
- check consistency with previous frame
- check consistency with prompt
You MUST explicitly note:
- any object appearing/disappearing
- any motion inconsistency
- any identity change

STEP 2: LOCAL ERROR DETECTION
For EACH frame, mark:
- "OK"
- "Minor issue"
- "Major issue"
- "Critical failure"
BUT only if there is visible evidence.

STEP 3: FRAME SCORE ASSIGNMENT RULE
Each frame_score[i] is determined ONLY by:
- 9-10: fully consistent frame, no visible issues
- 7-8: minor ambiguity or slight motion inconsistency
- 4-6: clear but not severe issue
- 1-3: strong inconsistency (wrong object, broken motion)
- 0: frame is visually incoherent or contradicts previous frame
IMPORTANT:
- Scores MUST reflect local evidence
- Do NOT reuse same score unless frames are truly identical

STEP 4: GLOBAL SCORE COMPUTATION
Global scores MUST be computed ONLY from:
- number of "Major/Critical" issues
- frequency of inconsistencies
- coverage of prompt actions
No direct subjective judgment allowed.

SCORING ANCHOR (VERY IMPORTANT):
- Perfect video: 9–10
- Mostly correct with minor noise: 7–8
- Noticeable errors (missing / wrong actions): 5–6
- Severe temporal/physical errors: 2–4
- Almost meaningless video: 0–1

OUTPUT FORMAT (STRICT JSON):

```json
{{
 "think": {{
    "frame_evidence": [
       "frame 1: ...",
       "frame 2: ...",
       "frame 3: ..."
    ],
    "detected_errors": [
       "..."
    ],
    "aggregation_logic": ""
 }},
 "frame_scores": [0, 0, 0, 0, 0, 0],
 "temporal_order_score": 0,
 "physical_plausibility_score": 0,
 "visual_consistency_score": 0,
 "prompt_alignment_score": 0,
 "final_score": 0
}}
```

Prompt: caption
Video: sampled frames

Figure 6: Prompt instructions for evaluating the Temporal-Following Score.

Table 6: Comprehensive hyperparameters for TempAct planner–executor reinforcement learning. We summarize the configurations used for the shared autoregressive video setup, planner RL, executor RL, hierarchical sampling, and streaming rollout.

Category	Setting	Value
Model & Video Specs	Backbone	Self-Forcing / LongLive
	Denoising Timesteps (T)	4 from $[1000, 750, 500, 250]$
	Local Attention Size	9 / 12
	Sink Size	1 / 3
	Local Size	8 / 9
	Frames per chunk	3
Planner RL	Planner Policy	LLM planner π_ϕ
	LoRA Fine-Tuning Rank (r)	32
	Scaling Factor (α)	64
	Dropout Rate	0.0
	Gradient Checkpointing	True
	Optimization Objective	GSPO
	Planning Group Size (M)	4
	Step Span (L)	6 latent frames
	Prompt Segments	5
	Reward Signals	plan judge + video-level reward
	Gradient Accumulation Steps	2
	Judge / Video Reward Weights	0.15 / 0.85
	KL Reference Policy	frozen initialization policy
	EMA Decay Rate	None
	KL Penalty Weight	5×10^{-4}
	Learning Rate	2×10^{-5}
	Advantage Clip Max	1
	Clip Range ($\epsilon_{\text{low}}, \epsilon_{\text{high}}$)	$(5 \times 10^{-4}, 5 \times 10^{-4})$
Executor RL	LoRA Fine-Tuning Rank (r)	256
	Scaling Factor (α)	256
	Executor Policy	AR diffusion executor p_θ
	Optimization Objective	Flow-GRPO
	Continuation Group Size (N)	8
	Reward Signals	local step-following + PickScore
	Step / Aesthetic Reward Weights	0.6 / 0.4
	Updated Region	first chunk after prompt switch
	Optimizer	AdamW
	Learning Rate (η)	1×10^{-5}
	LR Scheduler	constant_with_warmup
	Gradient Accumulation Steps	2
	Max Gradient Norm	1.0
	EMA Decay Rate	0.9 / 0.99
	KL Penalty Weight	5×10^{-4}
	Advantage Clip Max	2
	Clip Range ($\epsilon_{\text{low}}, \epsilon_{\text{high}}$)	$(1 \times 10^{-5}, 1 \times 10^{-5})$
Hierarchical RL	Training Transition Latent Frames (k)	random from $[6, 12]$
	Max Train Steps	400
	Training Hardware	32 NVIDIA H20 GPUs
Streaming Rollout	Target Latent Video Length (K)	30 latent frames
	Chunk Size	3 frames
	Prompt Switching	transition-aware prompt smoothing
	VAE Temporal Stride	4

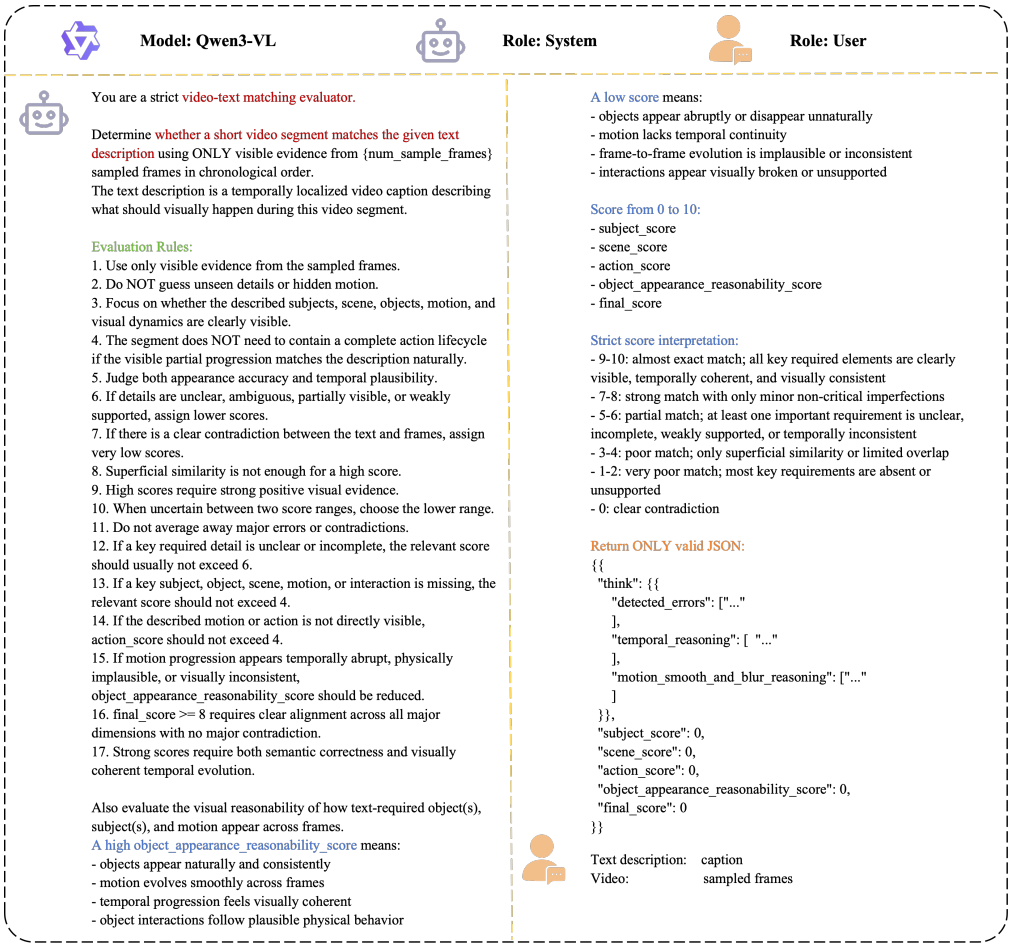
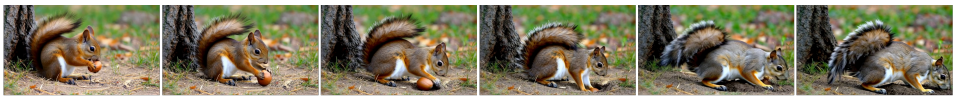


Figure 7: Prompt instructions for evaluating the Local Step-Following Score.



{**Think**: {
frame_evidence: [
frame 1: Squirrel is sitting on bare ground near tree root, holding an acorn with front paws, examining it closely. Consistent with prompt's first action.,
frame 2: Squirrel is still holding the acorn with front paws, head slightly lowered, continuing examination. Motion consistent with frame 1. No major change in action.,
frame 3: Squirrel is still holding the acorn, but now appears to be turning it over with paws. Motion consistent with examining. No major change.,
frame 4: Squirrel has released the acorn and is now holding it with one paw while the other paw is on the ground. Acorn is on the ground. Action transition begins.,
frame 5: Squirrel is leaning forward, paw near the acorn, beginning to scratch the ground. Acorn is still on the ground. Action transition to digging.,
frame 6: Squirrel is actively scratching the ground with front paws, creating a small hole. Acorn is still visible near the hole. Action consistent with digging.,
frame 7: Squirrel is still digging, with visible hole and displaced soil. Acorn is no longer visible, suggesting it has been dropped into the hole. Action consistent with prompt.,
frame 8: Squirrel is still digging, but now appears to be pushing dirt with its paws. Acorn is not visible, suggesting it has been buried. Action consistent with prompt.,
frame 9: Squirrel is still pushing dirt with paws, body low to the ground. Acorn is not visible. Action consistent with patting dirt.,
frame 10: Squirrel is still patting the ground with paws, body posture suggests completion of burying. Acorn is not visible. Action consistent with prompt.
],
detected_errors: [
In frames 1-4, the squirrel is examining the acorn, but the prompt says it should set it down before digging. The transition is not clearly shown as a distinct action; the acorn is still held in frame 4, which is ambiguous.,
In frame 7, the acorn is no longer visible, but the prompt says the squirrel drops it into the hole. The action of dropping is not visually confirmed; it is inferred from the acorn's disappearance.,
In frames 8-10, the squirrel is patting the ground, but the prompt says it 'pushes loose dirt back over it'. The motion is ambiguous whether it is patting or pushing, and the acorn's location is not visually confirmed to be covered.
],
aggregation_logic:
The video shows the squirrel performing the actions in sequence, but with ambiguous transitions and missing visual confirmation of key actions (dropping the acorn, covering it). The physical actions are plausible, but the prompt's specific sequence is not fully and explicitly verified in the frames. The video is mostly consistent but has minor ambiguities in action transitions and object state changes.
},
frame_scores: [9, 9, 9, 8, 8, 9, 9, 9, 9, 9],
temporal_order_score: 9,
physical_plausibility_score: 9,
visual_consistency_score: 9,
prompt_alignment_score: 7,
final_score: 8
}



{**Think**: {
frame_evidence: [
frame 1: Chef in white apron stands at wooden cutting board. No tomato or knife present. Background shows kitchen counter.
frame 2: Chef places a ripe tomato on the board and steadies it with left hand. Right hand holds a chef's knife poised to slice. Consistent with prompt.
frame 3: Chef begins slicing the tomato. Knife is in motion, cutting into the tomato. Left hand still steadies the tomato. Consistent with prompt.
frame 4: Tomato is sliced into three even rounds. Chef continues slicing. Slices are visible on board. Consistent with prompt.
frame 5: Chef is cutting a fourth slice. Knife is in motion. Slices are neatly arranged. Consistent with prompt.
frame 6: Chef has finished slicing. Left hand is now touching the slices. Knife is still in hand. Consistent with prompt.
frame 7: Chef has placed the sliced tomato onto a white plate. Plate is now visible on the cutting board. Consistent with prompt.
frame 8: Chef is using a small utensil (possibly a spoon) to adjust the tomato slices on the plate. Right hand is holding the utensil. Consistent with prompt.
frame 9: Chef continues adjusting the slices with the utensil. Left hand is near the plate. Consistent with prompt.
frame 10: Chef has finished arranging. Both hands are resting near the plate. Plate is full of neatly arranged tomato slices. Consistent with prompt.
],
detected_errors: [
No critical failures or major inconsistencies observed. All actions are visually consistent with the prompt and temporally ordered.
],
aggregation_logic:
All 10 frames are fully consistent with the prompt. No motion inconsistencies, object disappearances, or identity changes are visible. The sequence accurately depicts the chef placing the tomato, slicing it, and arranging the slices on a plate. All actions are physically plausible and temporally coherent. No errors warranting a score below 9.
},
frame_scores: [9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
temporal_order_score: 9,
physical_plausibility_score: 9,
visual_consistency_score: 9,
prompt_alignment_score: 9,
final_score: 9
}

Figure 8: Examples of Temporal-Following scoring results produced by the VLM evaluators.

F HIERARCHICAL SAMPLING AND CREDIT ASSIGNMENT ABLATIONS

The component ablations in Table 2 compare planner-only, executor-only, and joint optimization. To distinguish the contributions of sample organization and executor credit assignment from the benefit of jointly training both policies, we evaluate two additional controlled ablations: cross-plan executor advantages with full hierarchical sampling, and independent prefixes with within-plan executor advantages. Table 7 reports their results alongside the full-method scores from Table 2.

Common setup. Both ablations retain joint planner–executor training on Self-Forcing, with $M = 4$ candidate plans and $N = 8$ continuations per plan for each global instruction. They use the same initialization, training instructions, prompt smoothing, reward models and weights, KL regularization, optimizer settings, and number of policy updates as the full method. The executor update remains restricted to the first chunk after the selected prompt transition. The planner retains its plan-quality reward, continuation-averaged video reward, and plan-level advantage computation. Evaluation follows the Temporal Order benchmark protocol using the in-domain Qwen3-VL-8B judge; Avg. denotes the mean of the Simple and Hard scores. Each variant samples fresh rollouts from its own current policies during training.

F.1 FULL SAMPLING WITH CROSS-PLAN EXECUTOR ADVANTAGES

This variant preserves the complete hierarchical sampling procedure: each plan $\mathbf{s}^m$ has one shared prefix $\mathbf{x}^m_{1:k}$ and N sampled continuations. It changes only the normalization group used to compute executor advantages. Instead of normalizing within each plan–context group, it pools the MN executor rewards belonging to the same global instruction:

$$\begin{aligned} \mu_{\text{all}} &= \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} R^{m,n}_{\text{exec}}, \\ \sigma_{\text{all}} &= \text{std}\Big(\{R^{m,n}_{\text{exec}}\}^{M,N}_{m=1,n=1}\Big), \\ A^{m,n}_{\text{exec,cross}} &= \frac{R^{m,n}_{\text{exec}} - \mu_{\text{all}}}{\sigma_{\text{all}} + \epsilon}. \end{aligned} \tag{17}$$

All other terms in the executor objective are unchanged. In particular, each likelihood ratio is evaluated under the sample's own plan and visual history; pooling rewards does not exchange conditioning states. This comparison tests whether executor learning benefits from evaluating continuations relative to alternatives under the same plan and context, rather than relative to executions of different plans. Since the normalization group also grows from $N = 8$ to $MN = 32$, this contrast alone cannot separate conditioning-aware grouping from normalization-group size.

F.2 INDEPENDENT PREFIXES WITH WITHIN-PLAN EXECUTOR ADVANTAGES

This variant retains M plans and N executions per plan, but removes the shared-prefix constraint. For each plan, it independently generates a prefix for every continuation at the same selected transition index k:

$$\mathbf{x}^{m,n}_{1:k} \sim \prod_{i=1}^{k} p_\theta\Big(\mathbf{x}^{m,n}_i \mid \mathbf{x}^{m,n}_{<i}, \tilde{s}^m_{\sigma(i)}\Big), \qquad n = 1, \ldots, N. \tag{18}$$

Each continuation is then sampled from its own prefix $\mathbf{x}^{m,n}_{1:k}$. Executor rewards are still normalized over the N executions of the same plan:

$$A^{m,n}_{\text{exec,ind}} = \frac{R^{m,n}_{\text{exec}} - \mu^m_{\text{exec}}}{\sigma^m_{\text{exec}} + \epsilon}, \tag{19}$$

where the group statistics now include executions with different preceding histories. The reward functions and transition-chunk update rule remain unchanged, and likelihood ratios use each sample's actual prefix. Relative to the full method, this experiment tests the effect of fixing visual history when forming execution groups. It measures the system-level effect of prefix sharing: independently generated histories can also alter the video feedback received by the planner, even though its reward aggregation rule is unchanged.

Table 7: Ablations of hierarchical sampling and executor credit assignment on Self-Forcing, evaluated with Qwen3-VL-8B. Full TempAct scores are reproduced from Table 2. All variants jointly train the planner and executor with $M = 4$ and $N = 8$. Avg. is the mean of the Simple and Hard Temporal-Following Scores.

Setting	Prefixes per plan	Executor grouping	Simple↑	Hard↑	Avg.↑
Full TempAct (reported)	Shared	Within plan/context	0.500	0.336	0.418
Cross-plan advantages	Shared	Across plans	0.441	0.307	0.374
Independent prefixes	Independent	Within plan	0.435	0.302	0.369

Results and analysis. Table 7 shows that full TempAct outperforms both variants on the Simple and Hard sets. With the sampling procedure unchanged, cross-plan executor advantages reduce the average Temporal-Following Score from 0.418 to 0.374, with Simple and Hard scores decreasing from 0.500 to 0.441 and from 0.336 to 0.307, respectively. This supports within-plan/context normalization over cross-plan normalization under the tested settings, although the comparison also changes the normalization-group size. Using independent prefixes while retaining within-plan advantages reduces the average score to 0.369, with scores of 0.435 and 0.302 on the two subsets. This result supports sharing visual history when forming execution groups in the jointly trained system. Together, these ablations demonstrate the advantages of TempAct's conditional grouping and shared-prefix sampling strategies in improving temporal instruction following under joint planner–executor optimization.

G ADDITIONAL QUALITATIVE RESULTS

Figure 9 complements the Self-Forcing comparison in Figure 3 with temporally ordered prompts using LongLive. It compares single-prompt generation, step-prompt generation, and TempAct under the same prompts.

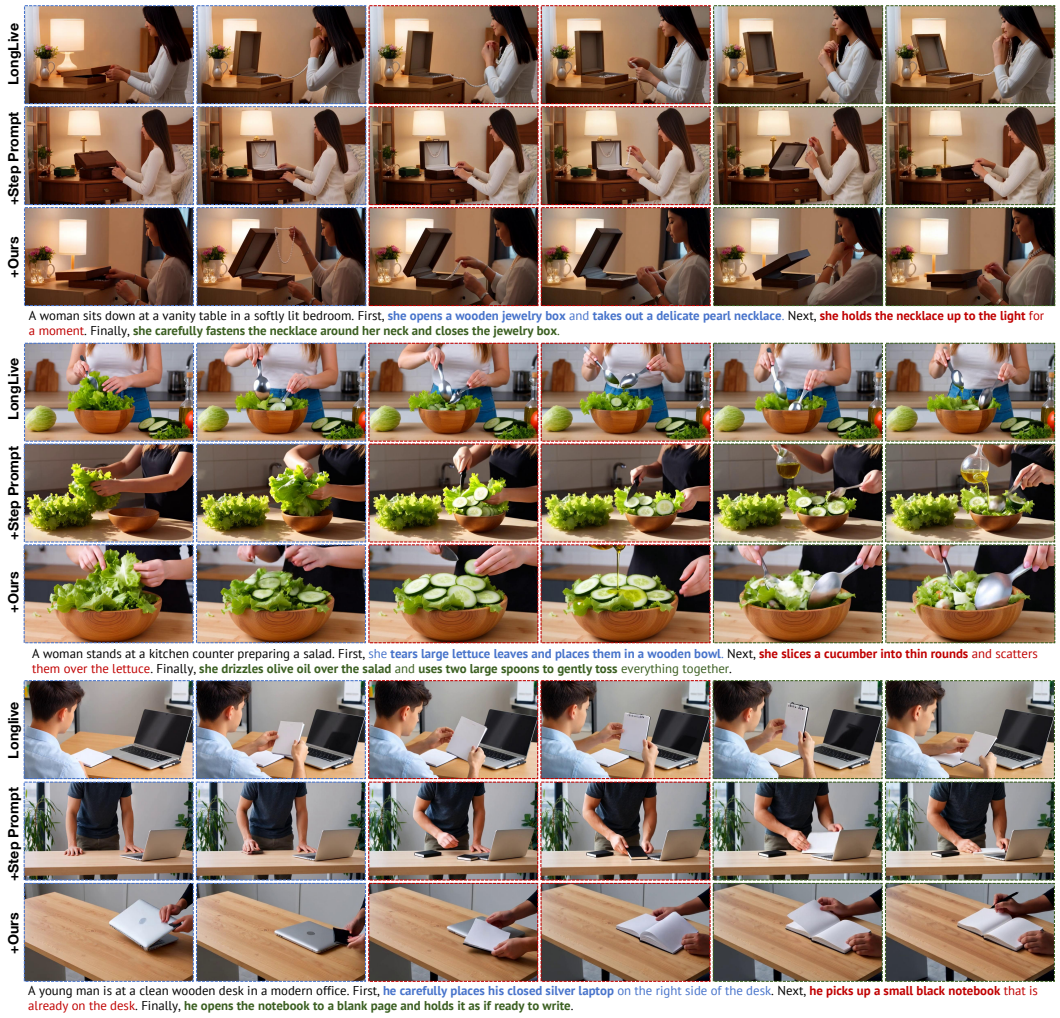


Figure 9: Qualitative comparison on temporally ordered prompts using the LongLive backbone.