TGUIDE: FASTER AUTOREGRESSIVE VIDEO DIFFUSION VIA HISTORICAL TRAJECTORY GUIDANCE

Anonymous authors
Paper under double-blind review

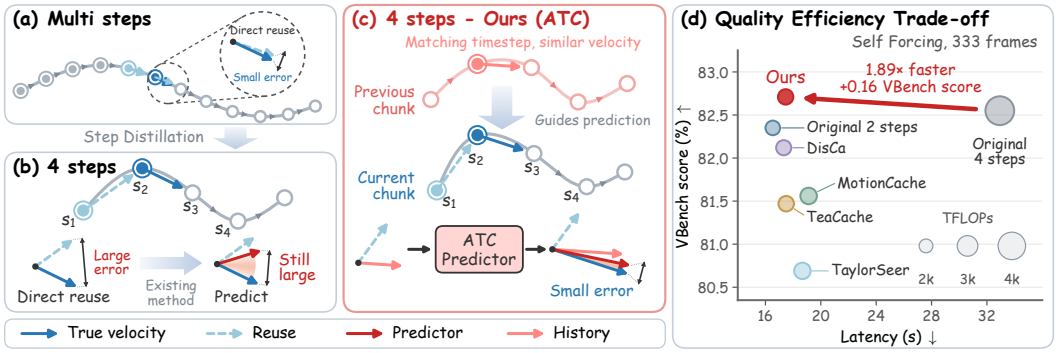


Figure 1: **Motivation and overview of TGuide.** Curves and arrows schematically illustrate sampling trajectories and velocity relationships. **(a)** With many denoising steps, nearby velocities can be similar, making direct reuse a useful approximation. **(b)** Step distillation increases the discrepancy between adjacent step velocities, leading to large errors in both direct reuse and existing prediction methods. **(c)** Past sampling trajectories guide current velocity prediction: ATC combines the previous chunk's velocity at the matching timestep with the current chunk's velocity from an earlier step to improve prediction accuracy in 4-step generation. **(d)** On Self Forcing with 333-frame generation, our TGuide achieves a $1.89\times$ speedup over the original 4-step baseline while improving the VBench score by 0.16 points. Bubble sizes indicate computational cost in TFLOPs.

ABSTRACT

Can autoregressive video diffusion be accelerated further when only four denoising steps remain? Existing feature caching approaches struggle in this regime, as large feature changes between denoising timesteps undermine both direct reuse and lightweight predictors. We observe that features across video chunks remain strongly correlated at matching denoising timesteps, offering a complementary source of guidance. Building on this insight, we introduce TGuide, a framework that accelerates autoregressive video diffusion through a lightweight Anchor-Transport-Correct (ATC) predictor guided by historical sampling trajectories. Our ATC combines two complementary references: the current chunk's preceding-step features preserve its spatial structure, while the previous chunk's same-timestep features provide information at the target denoising stage. By integrating these references through spatial retrieval and adaptive correction, ATC improves velocity prediction accuracy despite the large gaps between denoising steps. To mitigate error accumulation across both denoising steps and video chunks, we train the predictor on its own autoregressive rollouts, allowing it to adapt to states shaped by its own earlier predictions. An optional confidence-aware scheduler further adjusts the quality–efficiency trade-off. Experiments on Self Forcing, Causal Forcing, and HY-WorldPlay show that TGuide preserves generation quality while achieving denoising speedups of up to $1.89\times$ over the original 4-step generators. On Self Forcing, TGuide generalizes from 81-frame training videos to 333-frame generation, approximately four times the training length, while exceeding the original model's VBench score.

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.

1 INTRODUCTION

Autoregressive video diffusion enables continuous video generation by synthesizing successive frames or chunks conditioned on previously generated content (Chen et al., 2024a; Huang et al., 2025; Yin et al., 2025). Its sequential generation process allows user controls to guide subsequent content, supporting interactive applications and game simulation (He et al., 2025; Sun et al., 2025). To reduce the computational overhead of autoregressive generation, recent methods, including CausVid (Yin et al., 2025), Self Forcing (Huang et al., 2025), and Causal Forcing (Zhu et al., 2026), use distribution matching distillation (DMD) to reduce generation to just four denoising steps per chunk. However, each step still requires an expensive Diffusion Transformer (DiT) evaluation to predict the denoising velocity, making it difficult to satisfy the requirements of real-time generation. For example, on an NVIDIA A100 GPU, 4-step Self Forcing requires 32.9 seconds of denoising to generate a 20.8-second video (333 frames at 16 FPS; Tab. 1). This shortcoming motivates further acceleration of few-step autoregressive video generation.

Complementary to reducing the sampling steps, feature caching methods such as TeaCache (Liu et al., 2025a) exploit cross-timestep feature redundancy to reduce computation within each step, achieving substantial speedups under multi-step sampling schedules. However, 4-step sampling introduces larger gaps between adjacent denoising states, weakening the similarity that supports feature reuse and increasing the error of directly reusing cached features (Figs. 1(a,b) and 2(a)). To mitigate reuse errors, DisCa (Zou et al., 2026) learns a lightweight predictor to approximate target features from cached features, enabling effective caching in models distilled to 10–20 denoising steps. However, extending this approach to 4-step generation remains challenging: the predictor must approximate a large change in denoising stage with only a few DiT blocks, leading to reduced prediction accuracy (Fig. 1(b)). Motivated by feature correlations across video frames, C^3ache (Zhao et al., 2026) and X-Cache (Zeng et al., 2026) exploit redundancy along the video-time axis of autoregressive DiTs by reusing features from historical chunks. However, direct reuse across chunks does not account for motion-induced feature differences, and larger motion can lead to greater reuse errors. Together, these limitations raise a question: *can historical features still provide useful guidance for further acceleration, even when only four denoising steps remain?*

To investigate this possibility, we first examine feature correlations along both denoising time and video time using DiT final hidden states. As shown in Fig. 2(a), compared with the 50-step teacher references, the 4-step autoregressive generators exhibit substantially lower feature similarity across denoising timesteps, while similarity across video chunks at matching timesteps remains largely preserved. Moreover, oracle optical-flow alignment further increases cross-chunk similarity, suggesting that spatial displacement obscures useful historical correspondence. These observations suggest that historical features can remain informative even when they are unsuitable for direct reuse. In particular, the previous chunk provides a reference from its sampling trajectory at the target denoising stage, complementing the current chunk's preceding-step information tied to its current content (Fig. 1(c)). This motivates combining these complementary references to improve lightweight feature prediction in few-step autoregressive video generation.

Building on these insights, we introduce TGuide, a framework for accelerating few-step autoregressive video diffusion through historical trajectory guidance. At its core is an Anchor-Transport-Correct (ATC) predictor that uses the previous chunk's sampling trajectory to guide current velocity prediction. *Anchor* forms an initial prediction from the current noisy input and the current chunk's preceding-step features. To account for spatial displacement across chunks, *Transport* retrieves relevant information from the previous chunk's trajectory at the matching denoising timestep. *Correct* then selectively incorporates this information to refine the anchor prediction. ATC operates in DiT final hidden-state space, and the predicted state is decoded into velocity through the frozen output head. During autoregressive generation, approximate predictions can influence the states encountered in subsequent steps and chunks. We therefore train ATC through self-rollout on-policy distillation, allowing it to adapt to inputs shaped by its own earlier predictions. As an optional extension, a confidence-aware scheduler provides additional control over the quality–efficiency trade-off.

Experiments on Self Forcing, Causal Forcing, and HY-WorldPlay demonstrate that TGuide accelerates autoregressive video generation while preserving generation quality. Notably, despite training only on 81-frame videos, TGuide achieves a $1.89\times$ denoising speedup on 333-frame Self Forcing generation, approximately four times the training length, while surpassing the original 4-step

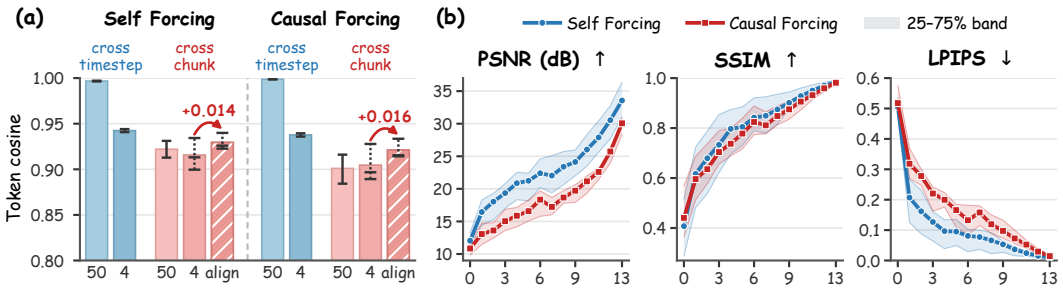


Figure 2: **(a) Correlation along denoising time and video time.** The 4-step autoregressive generators exhibit lower cosine similarity of final hidden states across denoising timesteps than the 50-step teacher references, while correlations across video chunks remain largely preserved. Optical-flow alignment further improves cross-chunk similarity. **(b) Global impact of local approximation.** Velocity reuse in a single chunk generally causes larger full-video deviations from a full-compute reference when applied earlier in the rollout, motivating more conservative approximation decisions for earlier chunks. The horizontal axis indicates the index of the chunk perturbed by velocity reuse. See Appendix E for diagnostic protocols.

model's VBench Score (Fig. 1(d)). Ablations further support the effectiveness of historical feature retrieval, correction, and self-rollout training. The contributions are summarized as follows:

- We propose TGuide, a framework for accelerating few-step autoregressive video diffusion through historical trajectory guidance. Its Anchor-Transport-Correct (ATC) predictor uses historical sampling trajectories to improve current velocity prediction through spatial retrieval and adaptive correction.
- We combine self-rollout on-policy distillation with an optional confidence-aware scheduler to support reliable autoregressive acceleration.
- Experiments across three autoregressive video generators demonstrate favorable quality–efficiency trade-offs and generalization to videos longer than those used for training. Ablations further support the effectiveness of our key designs.

2 Related Work

Autoregressive Video Diffusion. Autoregressive video diffusion enables continuous generation by synthesizing frames or chunks conditioned on historical context. Diffusion Forcing (Chen et al., 2024a) introduces flexible temporal noise modeling, subsequently extended to scalable chunk-wise video generation (Chen et al., 2025; Teng et al., 2025). CausVid, Self Forcing, and Causal Forcing advance few-step generation through distillation and improved training–inference alignment (Yin et al., 2025; Huang et al., 2025; Zhu et al., 2026), alongside complementary developments in adversarial training, reward-guided optimization, and temporal dependency modeling (Lin et al., 2025; Zhang et al., 2026; Lu et al., 2026b;a; Liu et al., 2026b). To sustain generation over longer horizons, LongLive (Yang et al., 2025) and related approaches improve context management and rollout training to mitigate temporal degradation (Zhang et al., 2025; Liu et al., 2026c; Cui et al., 2026; Chen et al., 2026; Yuan et al., 2026). HY-WorldPlay (Sun et al., 2025) demonstrates action-controlled interactive generation, a capability also explored by recent world models (Wang et al., 2026; Gao et al., 2026). Building on few-step autoregressive generators, our work reduces denoising computation through lightweight feature prediction guided by historical sampling trajectories.

Feature Caching for Diffusion Transformers. Feature caching accelerates diffusion transformers by reusing intermediate features across denoising timesteps along a sampling trajectory, with fixed or adaptive update schedules (Chen et al., 2024b; Selvaraju et al., 2024; Zhao et al., 2025; Zou et al., 2024; Liu et al., 2025a; Kahatapitiya et al., 2025; Ma et al., 2025; Bu et al., 2026), while forecasting and learning-based methods improve approximation through extrapolation, optimized caching policies, and learned feature transformations (Liu et al., 2025c; Zheng et al., 2025; Liu et al., 2025b; Zheng et al., 2026; Ma et al., 2024; Huang et al., 2024a; Liu et al., 2026a). DisCa (Zou et al., 2026) extends this direction with a lightweight predictor conditioned on cached features and

the current noisy input, enabling caching in step-distilled models. These approaches primarily exploit information within a single sampling trajectory. However, under 4-step sampling, the larger gaps between adjacent denoising states make both direct reuse and lightweight prediction along this trajectory more challenging. For autoregressive generation, recent methods adapt caching to chunk-dependent dynamics (Ma et al., 2026; Xu et al., 2026; Nan et al., 2026). C^3ache (Zhao et al., 2026) reuses the aggregate DiT residual across inference chunks at matching denoising steps, with scheduled cache refreshes. X-Cache (Zeng et al., 2026) performs block-level residual reuse with adaptive gating. Both methods directly reuse residuals from historical trajectories at corresponding denoising stages. However, matching denoising stages does not guarantee spatial correspondence: motion can displace historical content, while direct residual reuse does not explicitly recover these correspondences. In contrast, our ATC predictor retrieves and adapts information from historical sampling trajectories at the target denoising stage to guide current velocity prediction. This allows historical features to provide useful guidance even when direct reuse is unreliable, improving prediction accuracy at low computational cost.

3 METHOD

3.1 PRELIMINARIES

Autoregressive Video Diffusion. Videos are generated along two axes: video chunks j and denoising steps i within each chunk. Let x_i^j denote the noisy latent at step i of chunk j, and let $\mathcal{C}^j$ collect the external conditions and historical KV cache from earlier chunks. Starting from noise, each chunk is refined over N denoising steps at timesteps $t_0, \ldots, t_{N-1}$ conditioned on $\mathcal{C}^j$. Let $\mathcal{S}_i$ denote the sampling update at step i. Each step performs the update

$$x_{i+1}^j = \mathcal{S}_i\Big(x_i^j,\, v_\theta(x_i^j, t_i, \mathcal{C}^j)\Big)\,. \tag{1}$$

Here, $v_\theta(\cdot)$ is the velocity prediction from the Diffusion Transformer (DiT) with parameters θ.

Few Step Distillation. Generating K chunks requires $K \cdot N$ full DiT denoising evaluations. Step distillation trains a student generator to approximate a teacher's generation process using fewer denoising steps, thereby reducing N. This work considers already-distilled autoregressive generators, with $N = 4$; each of the four denoising steps ordinarily still requires a full DiT evaluation.

Feature Caching. One can also limit the computational cost of select DiT evaluations. Let $\mathcal{H}_i^j$ denote cached representations available at step i of chunk j. At selected steps, a reuse or prediction function $\mathcal{G}$ supplies an approximate velocity $\hat{v}_i^j$ in place of v_θ in Eq. 1:

$$\hat{v}_i^j = \mathcal{G}\Big(x_i^j, t_i, \mathcal{C}^j; \mathcal{H}_i^j\Big)\,. \tag{2}$$

Cached computation can approximate the target features through direct reuse, extrapolation, or a lightweight learned predictor. In few-step generators, larger gaps between denoising states make these cross step approximations less accurate, motivating more informative references.

3.2 PROBLEM SETUP AND MOTIVATION

We seek to accelerate the 4-step generator by replacing selected DiT evaluations with lightweight predictions. We first consider reusing computation across denoising steps within the same chunk, and then show how preceding chunks provide a complementary source of information.

Prediction from cached computation. Following Eq. 2, we use the cached velocity from an earlier denoising step of the current chunk as an anchor for prediction:

$$\hat{v}_{i,P}^j = \mathcal{P}\Big(x_i^j, t_i, \mathcal{C}^j; v_{i_a}^j\Big)\,. \tag{3}$$

Here $i_a < i$ denotes the anchor step, usually $i_a = i - 1$. Its cached velocity $v_{i_a}^j$ may come from a full DiT evaluation or an accepted prediction. Eq. 3 specializes the generic cache $\mathcal{H}_i^j$ in Eq. 2

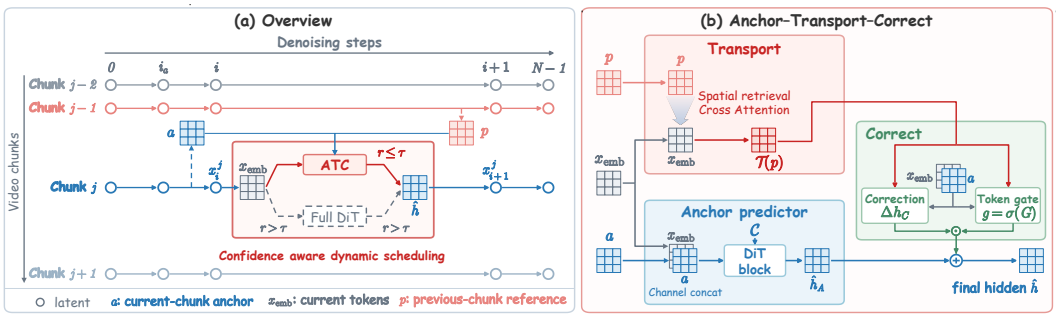


Figure 3: **Overview of TGuide. (a)** ATC replaces selected full DiT forward passes by predicting final hidden states from current tokens x_{emb}, the current chunk's anchor $a = h^j_{i_a}$, and the previous chunk's reference $p = h^{j-1}_i$ at the same timestep. Optional confidence-aware scheduling accepts predictions when $r \leq \tau$ and otherwise invokes the full DiT. **(b)** The anchor predictor estimates the current hidden state, Transport retrieves relevant historical features through cross-attention, and Correct applies a token-wise gated correction to produce $\hat{h}$.

to this reference. It is a natural starting point because the cached output and the target correspond to the same video content. However, with only four denoising steps, the cached output $v^j_{i_a}$ may differ substantially from the target output v^j_i because their timesteps are widely separated, consistent with the lower cross timestep feature similarity in Fig. 2(a). Consequently, the predictor bridges the denoising gap with substantially less computation than the full DiT, compromising prediction accuracy as illustrated in Fig. 1(b).

Guidance from the previous chunk. Our key observation is that, under 4-step autoregressive generation, feature similarity across chunks at matching timesteps remains comparable to that across denoising steps within a chunk (see Fig. 2(a)). We therefore augment the predictor with v^{j-1}_i, the cached velocity used at the target timestep of the previous chunk, to guide the current prediction as illustrated in Fig. 1(c):

$$\hat{v}^j_{i,\text{hist}} = \mathcal{P}_{\text{hist}}\left(x^j_i, t_i, \mathcal{C}^j; v^j_{i_a}, v^{j-1}_i\right). \tag{4}$$

Since the preceding chunk has already been processed, v^{j-1}_i is available during causal inference, whether it came from a full DiT evaluation or an accepted prediction. However, the previous chunk's output v^{j-1}_i cannot simply be copied, because motion and viewpoint changes may displace content or reveal new regions. Conversely, the current chunk's anchor output $v^j_{i_a}$ reflects its spatial structure but belongs to an earlier denoising stage. The two references therefore provide complementary information, motivating a predictor that adapts historical content to the current chunk while accounting for the change in denoising stage.

Method overview. As illustrated in Fig. 3, TGuide uses an Anchor-Transport-Correct (ATC) predictor to combine the current chunk's denoising anchor with historical information at the target timestep (Sec. 3.3). The ATC predictor first evolves the current chunk anchor towards the target denoising stage. It then retrieves relevant historical information across spatial positions, rather than assuming fixed position correspondence, and uses a token-wise gate to control the resulting correction. This decomposition lets historical computation refine the prediction where correspondence is useful, while preserving the anchor-based estimate where historical information is unreliable. An optional confidence-aware scheduler determines whether to accept each prediction or invoke the full DiT (Sec. 3.4), while training on self-rollouts prepares the predictor to consume its own approximations (Sec. 3.5).

3.3 Anchor-Transport-Correct Predictor

We now describe the ATC architecture for the history-guided predictor in Eq. 4. Although the formulation is expressed in velocity space, ATC predicts the DiT's final hidden state $\hat{h}$ in practice, which is then mapped to velocity through the frozen output head: $\hat{v} = \text{Head}(\hat{h}, t_i)$. For brevity,

let x_{emb} denote the current noisy latent x_i^j after the input head. We write $a = h_{i_a}^j$ for the current chunk's cached anchor, $p = h_i^{j-1}$ for the previous chunk's final hidden state at the same timestep, and $C = C^j$ for the current chunk's conditions and historical KV context. During inference, both a and p may come from either full evaluations or earlier accepted predictions. Historical KV and generation conditions are supplied to the predictor block as in the base model.

Anchor: evolve the current chunk along denoising steps. The cached anchor a retains the spatial layout of the current content, but represents an earlier denoising stage. We combine it with the current noisy tokens x_{emb} and the context $\mathcal{C}$ to predict a denoising increment:

$$\hat{h}_A = a + \Delta h_D\big(x_{\text{emb}},\, a,\, \mathcal{C}\big). \tag{5}$$

Here $\Delta h_D(\cdot)$ is implemented with a few DiT blocks, initialized with the weights from the full DiT.

Transport: retrieve history at the matching denoising stage. To account for motion and viewpoint changes between chunks, we use current tokens as queries to retrieve spatially corresponding information from the previous chunk's cached state p:

$$\mathcal{T}(p) = \text{CrossAttn}\big(Q(x_{\text{emb}}),\, K(p),\, V(p)\big). \tag{6}$$

The transported state $\mathcal{T}(p)$ contains historical features retrieved for the current tokens.

Correct: selectively refine the anchor evolution. The *Correct* branch combines transported history with the current inputs, using a scalar gate per token to control the correction:

$$\begin{aligned}\hat{h} &= \hat{h}_A + g \odot \Delta h_C\big(x_{\text{emb}},\, a,\, \mathcal{T}(p)\big) \\ &= a + \underbrace{\Delta h_D\big(x_{\text{emb}},\, a,\, \mathcal{C}\big)}_{\text{denoising evolution}} + \underbrace{g \odot \Delta h_C\big(x_{\text{emb}},\, a,\, \mathcal{T}(p)\big)}_{\text{gated cross chunk correction}}.\end{aligned} \tag{7}$$

Here, the lightweight MLP $\Delta h_C(\cdot)$ predicts a correction to $\hat{h}_A$ from the current tokens, anchor, and transported history. A network $G(\cdot)$ controls its strength through $g = \sigma(G(x_{\text{emb}}, a, \mathcal{T}(p)))$, where σ is the sigmoid function. The operator $\odot$ applies each token's scalar gate across feature channels.

Efficient Architecture Design. On Self Forcing, ATC uses a single DiT block and requires only 5.3% of the FLOPs and 6.8% of the latency of a full DiT forward pass. Unlike TaylorSeer (Liu et al., 2025c) and X-Cache (Zeng et al., 2026), which cache intermediate outputs for individual DiT blocks, ATC retains one final hidden state tensor per cached sampling position. This reduces the additional memory required for feature caching, which is particularly valuable for long video generation, where historical KV caches already consume substantial memory.

3.4 Confidence aware dynamic scheduling

Fig. 2(b) shows that introducing velocity reuse in earlier chunks generally leads to larger deviations over the full generated video. This observation motivates a scheduler that combines an estimate of local prediction error with a chunk-position prior, favoring more conservative acceptance decisions earlier in the rollout. While caching methods such as TeaCache (Liu et al., 2025a) rely on manually designed proxies to estimate reuse error, such proxies may not fully capture the complex feature dynamics of long-video generation. We instead attach a lightweight confidence head to ATC, learning to estimate its relative hidden-state error $\hat{\epsilon}_{j,i}$ from predictor features and sampling conditions.

We construct a scheduling risk score by weighting the estimated local error with the outgoing denoising span and a chunk-position prior:

$$r_{j,i} = \hat{\epsilon}_{j,i}\, \frac{|t_i - t_{i+1}|}{T}\big(1 + \beta\alpha_j\big), \tag{8}$$

where T normalizes the timestep span, α_j decreases linearly from one to zero across predictor-eligible chunks, and $\beta \geq 0$ controls the additional weight assigned to earlier chunks. Given a risk threshold τ, we accept the prediction when $r_{j,i} \leq \tau$ and otherwise invoke the full DiT. Confidence-head details and prior definitions are provided in Appendix C.

3.5 TWO STAGE TRAINING

Stage 1: Offline regression with teacher forcing. We first train ATC offline on trajectories collected from the frozen full DiT. Both the anchor hidden states and the previous chunk's same-timestep hidden states are taken from these trajectories, and ATC is supervised to match the full DiT's final hidden states and corresponding velocity predictions. This stage provides an initialization for subsequent self-rollout training.

Stage 2: Self-rollout on-policy distillation. During inference, however, ATC operates on states affected by its own imperfect predictions. These prediction errors propagate to subsequent noisy latents and cached features, creating a train-test distribution gap along both the denoising and chunk axes. To reduce this mismatch, we follow the self-rollout principle of Self Forcing (Huang et al., 2025) and perform autoregressive rollouts with ATC during training. Its predictions drive sampling updates and populate the feature cache, while generated chunks provide context for subsequent chunks. We fine-tune ATC on these self-generated trajectories while keeping the base generator frozen. Detailed training objectives and optimization settings are provided in Appendix D.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

Baselines and settings. We evaluate TGuide on three models: Self Forcing, Causal Forcing, and HY-WorldPlay. We compare with the original 4-step and 2-step models, TeaCache (Liu et al., 2025a), TaylorSeer (Liu et al., 2025c), MotionCache (Xu et al., 2026), and DisCa (Zou et al., 2026). Schedules use `F` for full DiT evaluation, `P` for prediction, and `R` for velocity reuse. The first chunk and the first denoising step of each subsequent chunk use full DiT. All experiments use NVIDIA A100 GPUs at 832×480 resolution, with TGuide using a fixed `FPPF` schedule in the main comparisons. Training and baseline implementation details are provided in Appendices D and A.4.

Evaluation protocol. We measure efficiency using TFLOPs, latency, and speedup over the original 4-step model. We report Quality, Semantic, and Total Scores. We use the full VBench suite (Huang et al., 2024b) for the main Self Forcing comparison and the ATC and self-rollout on-policy distillation ablations. Additional Causal Forcing results and dynamic scheduling ablations are provided in Appendices B.1 and C, respectively. For camera-conditioned image-to-video generation, we evaluate the generated videos using a subset of the VBench I2V suite. We follow HY-WorldPlay's camera evaluation protocol, using four cyclic camera trajectories that revisit previously observed views. PSNR, SSIM, and LPIPS are computed following HY-WorldPlay's protocol, including comparisons against the original 4-step outputs and under the self-comparison protocol, which compares corresponding views along the return and initial paths. Further evaluation details are provided in Appendix A.

4.2 QUALITATIVE COMPARISON

Fig. 4 illustrates how acceleration affects generated content over time. In the Self Forcing example, all competing acceleration methods exhibit changes in vehicle appearance or distortions in body structure by 20 s, whereas TGuide better preserves the vehicle's visual identity and structure. The Causal Forcing example shows more pronounced degradation: several competing methods produce extensive colorful artifacts at 20 s, while TGuide maintains a cleaner depiction of the player and court, with fewer conspicuous artifacts than even the original model in this example. For HY-WorldPlay, the enlarged views at 3 s show that TGuide better preserves facial contours under camera motion than the competing acceleration methods. Together, these examples demonstrate improved preservation of subject structure and local detail, supporting visual consistency over extended generation. Additional visualizations are provided in Appendix B.3.

4.3 QUANTITATIVE EVALUATION

Text-to-video generation. TGuide preserves generation quality while substantially reducing denoising computation. On Self Forcing, TGuide achieves a Total Score of 84.23 at a $1.80\times$ speedup

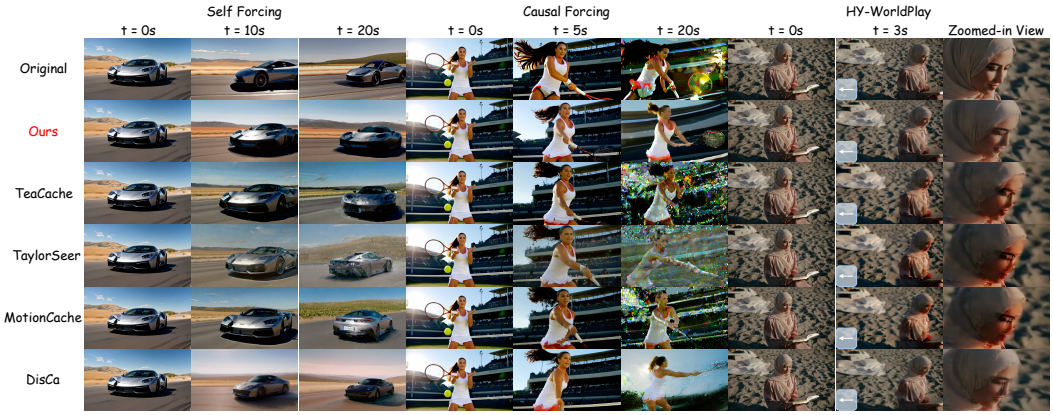


Figure 4: Qualitative comparisons on Self Forcing, Causal Forcing, and HY-WorldPlay.

Table 1: Performance comparison on Self Forcing using the full VBench benchmark.

Method	81 frames					333 frames				
	Latency (s) ↓	Speedup ↑	Total ↑	Quality ↑	Semantic ↑	Latency (s) ↓	Speedup ↑	Total ↑	Quality ↑	Semantic ↑
Original 4 steps	6.53	1.00×	84.28	84.63	82.87	32.93	1.00×	82.55	82.93	81.04
Original 2 steps	3.48	**1.88**×	84.09	84.46	82.59	16.53	**1.99**×	82.35	82.76	80.71
TeaCache	3.83	1.71×	83.91	84.14	83.01	17.50	1.88×	81.47	81.82	80.06
TaylorSeer	3.78	1.73×	83.68	83.93	82.69	18.67	1.76×	80.69	81.05	79.26
MotionCache	3.96	1.65×	83.97	84.25	82.85	19.12	1.72×	81.56	81.89	80.24
DisCa	3.59	1.82×	83.86	84.17	82.62	17.32	1.90×	82.12	82.44	80.84
Ours	3.62	1.80×	**84.23**	84.59	82.80	17.46	1.89×	**82.71**	83.12	81.06

for 81-frame generation, closely matching the original 4-step model's 84.28 (Tab. 1). At comparable latency, it outperforms DisCa by 0.37 points and the original 2-step model by 0.14 points, demonstrating a favorable quality–efficiency tradeoff under 4-step sampling. Although both training stages use only 81-frame videos, TGuide maintains its advantage at 333 frames, approximately four times the training length. It achieves a Total Score of 82.71 at a 1.89× speedup, surpassing DisCa by 0.59 points at comparable latency and even exceeding the original 4-step model's 82.55. These results suggest that historical guidance and self-rollout training help maintain generation quality when generalizing to substantially longer videos. Detailed 16-dimension Self Forcing scores appear in Tab. 6; Causal Forcing shows a similar advantage (Appendix B.1, Tab. 5).

Camera-conditioned image-to-video generation. TGuide achieves a 1.73× speedup while improving both fidelity to the original generator and consistency when revisiting previously generated content (Tab. 2). Against the original 4-step outputs, it achieves the best PSNR, SSIM, and LPIPS among accelerated methods. It also leads these methods under self-comparison, where frames on the return path are compared with corresponding frames from the initial pass. These complementary results suggest that the gains extend beyond matching the original outputs to preserving content across camera motion, consistent with our design of spatial retrieval and selective historical correction to adapt previous content to the current view.

4.4 Ablation Studies

We conduct all ablation studies on Self Forcing. We first compare predictor architectures under a common offline training protocol, and then evaluate the additional benefit of self-rollout on-policy distillation with the fixed ATC architecture.

Anchor-Transport-Correct ablation. Tab. 3 evaluates how historical trajectories guide prediction on 81-frame videos. Here, p is the previous chunk's cached final hidden state at the same denoising timestep (Sec. 3.3). All learned variants use the same Stage 1 offline regression protocol and fixed `FPPF` schedule. `FRRF` serves as the direct reuse baseline, while *w/o* p uses only the anchor predictor with causal KV context. Replacing direct reuse with learned prediction improves the Total Score from 83.14 to 83.86, showing the benefit of adapting cached features to the current denoising state.

Table 2: Performance comparison on HY-WorldPlay at 125 frames.

Method	Quality and efficiency			vs. Original 4 steps			Self-comparison		
	VBench ↑	Latency (s) ↓	Speedup ↑	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Original 4 steps	84.26	49.61	1.00×	—	—	—	16.75	0.5107	0.2820
Original 2 steps	84.18	26.81	**1.85**×	19.72	0.7122	0.1480	16.68	0.5124	0.2963
TeaCache	84.20	30.06	1.65×	18.63	0.7230	0.1413	16.44	0.5088	0.3026
TaylorSeer	<u>84.21</u>	29.76	1.67×	18.60	0.6845	0.1670	16.60	0.5128	0.2983
MotionCache	**84.24**	30.02	1.65×	18.59	0.6843	0.1672	16.60	0.5135	0.2986
DisCa	84.17	28.39	<u>1.75</u>×	19.87	0.7293	0.1296	16.81	0.5091	0.2955
Ours	**84.24**	28.75	<u>1.73</u>×	**20.52**	**0.7611**	**0.1144**	**16.87**	**0.5136**	**0.2872**

Concat p directly concatenates historical hidden states, current tokens, and the anchor along the channel dimension at matching spatial positions, improving the score to 83.98. *Anchor + Transport* retrieves historical features without the correction module and raises the score to 84.05, suggesting that spatial retrieval makes history more useful than direct concatenation. Finally, ATC adds the correction module and reaches 84.08, supporting the benefit of selectively refining the anchor prediction with retrieved history. Together, retrieval and correction improve over the predictor without p by 0.22 points with only 0.03 s of additional latency, demonstrating the value of adapting historical content beyond the context already supplied by causal KV.

Table 3: ATC ablation under Stage 1 training.

Method	Total ↑	TFLOPs ↓	Latency (s) ↓	Speedup ↑
Original 4 steps	84.28	793	6.53	1.00×
`FRRF`	83.14	429	3.42	1.91×
w/o p	83.86	444	3.59	1.82×
Concat p	83.98	445	3.59	1.82×
Anchor + Transport	84.05	448	3.60	1.81×
ATC	**84.08**	452	3.62	1.80×

Confidence aware dynamic scheduling. We treat dynamic scheduling as an optional extension and use `FPPF` in the main experiments to facilitate comparison with fixed strategies such as original 2-step and DisCa. Dynamic scheduling provides a flexible tradeoff between generation quality and efficiency; a suitable confidence threshold achieves better quality at lower latency than the fixed `FPFF` and `FFPF` schedules. Additional comparisons and detailed scheduling ablations are provided in Appendices B.1 and C, respectively.

Self-rollout on-policy distillation. In Tab. 4, Stage 1 uses offline teacher forcing, while Stage 2 applies self-rollout on-policy distillation. The results show that Stage 2 improves video generation quality, particularly at 333 frames, demonstrating generalization to longer videos despite training only on 81-frame sequences.

Table 4: Effect of Stage 2 training.

Predictor	81 frames		333 frames	
	Total ↑	Speedup ↑	Total ↑	Speedup ↑
Original 4 steps	84.28	1.00×	82.55	1.00×
Stage 1	84.08	1.80×	82.29	1.89×
+ Stage 2	**84.23**	1.80×	**82.71**	1.89×

5 Conclusion

We presented TGuide, a framework for accelerating 4-step autoregressive video diffusion through historical trajectory guidance. Our key insight is that historical features can remain informative for prediction even when they are unsuitable for direct reuse. The ATC predictor retrieves relevant information from the previous chunk's sampling trajectory and selectively incorporates it to refine the current anchor prediction. Self-rollout training adapts the predictor to states affected by its own approximations, while optional confidence-aware scheduling enables flexible quality–efficiency trade-offs. Experiments on Self Forcing, Causal Forcing, and HY-WorldPlay demonstrate the effectiveness of this approach across text-to-video and camera-conditioned generation. Notably, on Self Forcing, TGuide generalizes from 81-frame training videos to 333-frame generation, approximately four times the training length, achieving a $1.89\times$ denoising speedup while exceeding the original 4-step model's VBench Score. These results support historical trajectory guidance as an effective approach to accelerating autoregressive generation over extended rollouts.

AI USE STATEMENT

Generative AI tools were used in this work to assist with translation, language editing, grammar correction, and improving the clarity and conciseness of the manuscript. They were also used to support literature retrieval and summarization, and to assist with software development, including code generation, modification, and debugging.

Generative AI tools were not used to develop the research methodology, design experiments, formulate or prove mathematical claims, generate synthetic data, or draft sections of the paper from scratch. All AI-assisted text, literature-related information, and code were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REFERENCES

Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. Dicache: Let diffusion model determine its own cache. In *International Conference on Learning Representations*, volume 2026, pp. 73778–73803, 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, 2024a.

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.

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Δ-DiT: A training-free acceleration method tailored for diffusion transformers, 2024b. URL https://arxiv.org/abs/2406.01125.

Shuo Chen, Cong Wei, Sun Sun, Ping Nie, Kai Zhou, Ge Zhang, Ming-Hsuan Yang, and Wenhu Chen. Context forcing: Consistent autoregressive video generation with long context. *arXiv preprint arXiv:2602.06028*, 2026.

Jiaxing 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. In *International Conference on Learning Representations*, volume 2026, pp. 85802–85822, 2026.

Weilun Feng, Chuanguang Yang, Haotong Qin, Xiangqi Li, Yu Wang, Zhulin An, Libo Huang, Boyu Diao, Zixiang Zhao, Yongjun Xu, et al. Q-vdit: Towards accurate quantization and distillation of video-generation diffusion transformers. *arXiv preprint arXiv:2505.22167*, 2025.

Zelin Gao, Qiuyu Wang, Jiapeng Zhu, Jingye Chen, Zichen Liu, Qingyan Bai, Jiahao Wang, Yufeng Yuan, Hanlin Wang, Yichong Lu, et al. Infinite worlds with versatile interactions. *arXiv preprint arXiv:2607.07534*, 2026.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, et al. Matrix-game 2.0: An open-source real-time and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

Yushi Huang, Zining Wang, Ruihao Gong, Jing Liu, Xinjie Zhang, Jinyang Guo, Xianglong Liu, and Jun Zhang. Harmonica: Harmonizing training and inference for better feature caching in diffusion transformer acceleration. *arXiv preprint arXiv:2410.01723*, 2024a.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024b.

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.

Shanchuan Lin, Ceyuan Yang, Hao He, Jianwen Jiang, Yuxi Ren, Xin Xia, Yang Zhao, Xuefeng Xiao, and Lu Jiang. Autoregressive adversarial post-training for real-time interactive video generation, 2025. URL https://arxiv.org/abs/2506.09350.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363. IEEE, 2025a.

Jiacheng Liu, Peiliang Cai, Qinming Zhou, Yuqi Lin, Deyang Kong, Benhao Huang, Yupei Pan, Haowen Xu, Chang Zou, Junshu Tang, et al. Freqca: Accelerating diffusion models via frequency-aware caching. *arXiv preprint arXiv:2510.08669*, 2025b.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863. IEEE, 2025c.

Jinshan Liu, Haoran Qin, Xiaobing Tu, Jiacheng Liu, Jiahui Hu, Zhengan Yan, Yukun Xie, Kerui Shen, Jinkui Ren, Yuqi Lin, Xiantao Zhang, and Linfeng Zhang. Linca: Accelerating diffusion models via learnable decomposed feature caching, 2026a. URL https://arxiv.org/abs/2608.17973.

Jinxiu Liu, Xuanming Liu, Kangfu Mei, Yandong Wen, Ming-Hsuan Yang, and Weiyang Liu. Streaming autoregressive video generation via diagonal distillation. *arXiv preprint arXiv:2603.09488*, 2026b.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations*, volume 2026, pp. 91177–91196, 2026c.

Yanzuo Lu, Ronglai Zuo, and Jiankang Deng. Raven: Real-time autoregressive video extrapolation with consistency-model grpo. *arXiv preprint arXiv:2605.15190*, 2026a.

Yunhong Lu, Yanhong Zeng, Haobo Li, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jiapeng Zhu, Hengyuan Cao, Zhipeng Zhang, Xing Zhu, et al. Reward forcing: Efficient streaming video generation with rewarded distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34385–34397, 2026b.

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, 2024.

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*, 2026.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. Magcache: Fast video generation with magnitude-aware cache, 2025. URL https://arxiv.org/abs/2506.09045.

Kepan Nan, Wangbo Zhao, Penghao Zhou, Jun Li, Zhenheng Yang, Jian Yang, and Ying Tai. Accelerating autoregressive video diffusion via history-guided cache and residual correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43740–43750, 2026.

Weiming Ren, Huan Yang, Ge Zhang, Cong Wei, Xinrun Du, Wenhao Huang, and Wenhu Chen. Consisti2v: Enhancing visual consistency for image-to-video generation. *arXiv preprint arXiv:2402.04324*, 2024.

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

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. *arXiv preprint arXiv:2512.14614*, 2025.

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.

Wenhao Wang and Yi Yang. Vidprom: A million-scale real prompt-gallery dataset for text-to-video diffusion models. *Advances in Neural Information Processing Systems*, 37:65618–65642, 2024.

Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, et al. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory. *arXiv preprint arXiv:2604.08995*, 2026.

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.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6613–6623. IEEE, 2024b.

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.

Shenghai Yuan, Yuanyang Yin, Zongjian Li, Xinwei Huang, Xiao Yang, and Li Yuan. Helios: Real real-time long video generation model. *arXiv preprint arXiv:2603.04379*, 2026.

Yixiao Zeng, Jianlei Zheng, Chaoda Zheng, Shijia Chen, Mingdian Liu, Tongping Liu, Tengwei Luo, Yu Zhang, Boyang Wang, Linkun Xu, et al. X-cache: Cross-chunk block caching for few-step autoregressive world models inference. *arXiv preprint arXiv:2604.20289*, 2026.

Jingran Zhang, Ning Li, Yuanhao Ban, Andrew Bai, and Justin Cui. Reward-forcing: Autoregressive video generation with reward feedback. *arXiv preprint arXiv:2601.16933*, 2026.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models, 2025. URL https://arxiv.org/abs/2504.12626.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Enshu Liu, Rui Wan, Widyadewi Soedarmadji, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, et al. Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation. *arXiv preprint arXiv:2406.02540*, 2024.

Weisen Zhao, Lam Nguyen, Zhicong Lu, and Yuzhang Shang. C^{3}ache: Accelerating world action models with cross inference chunk cache, 2026. URL https://arxiv.org/abs/2606.08962.

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.

Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers, 2025. URL https://arxiv.org/abs/2508.16211.

Shikang Zheng, Guantao Chen, Qinming Zhou, Yuqi Lin, Lixuan He, Chang Zou, Peiliang Cai, Jiacheng Liu, and Linfeng Zhang. Let features decide their own solvers: Hybrid feature caching for diffusion transformers. In *International Conference on Learning Representations*, volume 2026, pp. 62767–62790, 2026.

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.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024.

Chang Zou, Changlin Li, Yang Li, Patrol Li, Jianbing Wu, Xiao He, Songtao Liu, Zhao Zhong, Kailin Huang, and Linfeng Zhang. Disca: Accelerating video diffusion transformers with distillation-compatible learnable feature caching. *arXiv preprint arXiv:2602.05449*, 2026.

APPENDIX

A Evaluation Details **15**

A.1 Text-to-video evaluation 15

A.2 Image-to-video evaluation 15

A.3 Efficiency metrics 16

A.4 Comparison methods 16

B Additional quantitative and qualitative results **17**

B.1 Additional quantitative comparisons 17

B.2 Full VBench results on Self Forcing 17

B.3 Additional qualitative results 18

C Confidence-aware dynamic scheduling **20**

D Training details **21**

D.1 Stage 1: offline feature regression 21

D.2 Stage 2: self-rollout on-policy distillation 21

D.3 Dataset 22

E Details of the Motivating Experiments **23**

F Predictor block selection on Self Forcing **23**

F.1 Source layer and initialization 23

A Evaluation Details

A.1 Text-to-video evaluation

For quantitative text-to-video evaluation, we evaluate Self Forcing at 81 and 333 decoded frames and Causal Forcing at 81 decoded frames, both at 832×480 resolution and 16 FPS. For text-to-video evaluation, we report VBench scores (Huang et al., 2024b) using the LLM-extended evaluation prompts released by Self Forcing (Huang et al., 2025). The same extended prompts are used both to condition video generation and as the text references for text-dependent VBench scoring.

Full VBench evaluation. The main Self Forcing comparison and the ATC and self-rollout on-policy distillation ablations (Tabs. 1, 3, and 4) evaluate all 16 VBench dimensions. We first normalize each dimension score using the official VBench normalization ranges. The Quality Score is the weighted average of Subject Consistency, Background Consistency, Temporal Flickering, Motion Smoothness, Dynamic Degree, Aesthetic Quality, and Imaging Quality, assigning weight 0.5 to Dynamic Degree and 1 to each other dimension (total weight 6.5). The Semantic Score averages Object Class, Multiple Objects, Human Action, Color, Spatial Relationship, Scene, Appearance Style, Temporal Style, and Overall Consistency with equal weights. The Total Score combines the Quality and Semantic Scores with weights of $4{:}1$. We report all three aggregate scores for the main comparison and the Total Score for the two ablations, expressed as percentages.

VBench subset evaluation. We evaluate confidence-aware dynamic scheduling through both an ablation (Tab. 7) and an additional comparison with competing acceleration methods on Self Forcing (Tab. 5). These experiments, together with the Causal Forcing comparison in Tab. 5, use VBench-251, a subset of eight VBench dimensions evaluated using 251 extended prompts, following ConsistI2V (Ren et al., 2024), ViDiT-Q (Zhao et al., 2024), and Q-VDiT (Feng et al., 2025). We generate one video per prompt. Specifically, 72 prompts from the Subject Consistency suite evaluate Subject Consistency, Motion Smoothness, and Dynamic Degree; 93 prompts from the Overall Consistency suite evaluate Overall Consistency, Aesthetic Quality, and Imaging Quality; and 86 prompts from the Scene suite evaluate Scene and Background Consistency. After normalizing the dimension scores using the official VBench normalization ranges, we compute a quality score as the weighted average of Subject Consistency, Background Consistency, Motion Smoothness, Dynamic Degree, Aesthetic Quality, and Imaging Quality, assigning a weight of 0.5 to Dynamic Degree and 1 to each other dimension. We average Scene and Overall Consistency to obtain a semantic score, and combine the quality and semantic scores with weights of $4{:}1$ to obtain the reported subset score as a percentage. Subset scores are compared only within this evaluation protocol and are not directly comparable to the full VBench scores.

A.2 Image-to-video evaluation

HY-WorldPlay is evaluated at 125 decoded frames at 832×480 resolution and 24 FPS. For camera-conditioned HY-WorldPlay, we randomly select 50 prompts with their corresponding initial images from the VBench I2V set, excluding all images and prompts used for training. Following the VBench-based evaluation practice of LongLive (Yang et al., 2025), we evaluate a subset of five dimensions: Subject Consistency, Background Consistency, Motion Smoothness, Aesthetic Quality, and Imaging Quality, and report the arithmetic mean of their scores. Following WorldPlay (Sun et al., 2025), we use the four cyclic camera trajectories described in the Dataset subsection (Appendix D.3) to revisit previously observed views.

We additionally evaluate PSNR, SSIM, and LPIPS in two settings. For comparison against the original 4-step model, we compare generated frames with the corresponding 4-step outputs under matched initial images, text conditions, camera trajectories, and random seeds. For self-comparison, each model first generates frames along a camera trajectory and then retraces the same path. We compare frames generated on the return path with the corresponding frames from the initial pass within that model's own generated sequence, following WorldPlay. The former measures fidelity to the original model's outputs, while the latter measures consistency when revisiting the same views.

A.3 EFFICIENCY METRICS

All methods receive identical input conditions within each base model. Efficiency is measured by average computation per video in TFLOPs (10^{12} floating-point operations), model-path latency per video, and speedup over the corresponding full baseline. TFLOPs and latency measure the computational cost and runtime of DiT denoising, respectively, excluding KV-cache refresh, VAE decoding, and text encoding. Both measurements include rejected predictor attempts as well as their full-evaluation fallbacks. Timings use matched hardware and execution conditions; speedup is the ratio of mean baseline latency to mean method latency.

A.4 COMPARISON METHODS

We implement all comparison methods within the corresponding base model and autoregressive generation pipeline of Self Forcing, Causal Forcing, and HY-WorldPlay. Within each base model, all methods use identical evaluation prompts, random seeds, and video generation settings. At the time of our experiments, official implementations of C^{3}ache (Zhao et al., 2026) and X-Cache (Zeng et al., 2026) were not publicly available, so we do not include these methods in our empirical comparisons. The first chunk always uses full 4-step denoising, while subsequent chunks follow each method's acceleration strategy and retain the original cross-chunk causal KV cache. Below, F denotes a full DiT evaluation, R denotes direct cache reuse, and P denotes a learned predictor replacing a full DiT evaluation.

Original 4-step (`FFFF`). The baseline uses the original 4-step denoising configuration, with a full DiT evaluation at every step, as the reference for generation quality and inference efficiency. The raw timesteps are $[1000, 750, 500, 250]$. All configurations use the base model's timestep warping with shift 5, $t_{\text{warp}} = 5t/(1 + 4t/1000)$, giving baseline timesteps of $[1000, 937.50, 833.33, 625.00]$ after rounding.

Original 3/2-step. These baselines reduce the number of denoising steps in each subsequent chunk to three or two and reset the sampling timesteps accordingly. For three-step sampling, Self Forcing and Causal Forcing use integer timesteps $[1000, 666, 333]$, which map to approximately $[1000, 908.84, 713.98]$. HY-WorldPlay instead uses unrounded, evenly spaced timesteps $[1000, 2000/3, 1000/3]$, which map to approximately $[1000, 909.09, 714.29]$. The two-step configuration uses $[1000, 500]$ for all three bases, mapping to approximately $[1000, 833.33]$. Every retained step executes the full DiT, with no additional training, to assess the quality–efficiency trade-off of directly reducing sampling steps.

Naive Cache 2-step (`FRRF`). This baseline is used only in the ablation study in Tab. 3. This baseline retains the original 4-step sampling schedule but evaluates the full DiT only at the first and fourth steps. The two middle steps reuse the velocity prediction cached at the first step to perform their respective sampling updates. Thus, each subsequent chunk still receives four denoising updates but requires only two full DiT evaluations, without feature prediction or adaptive cache decisions.

TeaCache. Our implementation of TeaCache (Liu et al., 2025a) uses the accumulated, rescaled relative change in timestep-modulated input features as a proxy for cache reuse error. When this quantity is below a threshold, it reuses the complete DiT block stack's feature residual from the most recent full evaluation; otherwise, it executes the full stack and refreshes the cache. The output head is evaluated at the current timestep. To accommodate autoregressive generation with repeated timestep schedules across chunks, the change indicator uses modulated input features containing the current video content, allowing cache decisions to respond to changes in chunk content. Base-specific calibration coefficients and cache thresholds are used, with the threshold controlling the quality–efficiency trade-off. Each chunk independently determines its compute/reuse schedule from its own content-dependent feature changes. This adaptation largely follows the chunk-wise independent caching principle of FlowCache (Ma et al., 2026), so we do not include a separate FlowCache comparison; our implementation retains the TeaCache decision criterion described above.

TaylorSeer. Our implementation of TaylorSeer (Liu et al., 2025c) caches attention and feed-forward outputs within each DiT block and estimates their changes across denoising steps using

finite differences at full evaluations. Skipped computations are replaced by Taylor extrapolation of these outputs, while retaining the current timestep modulation and residual connections. Feature histories are reset at the beginning of every chunk. We use at most first-order prediction: two available full evaluations enable a first-order difference estimate, while a single full evaluation permits only zeroth-order reuse. The refresh interval controls computational cost.

MotionCache. Our implementation of MotionCache (Xu et al., 2026) makes cache decisions at token granularity, estimating motion from feature differences between adjacent frames and using these estimates to weight each token's accumulated feature change. At chunk boundaries, the previous chunk's last frame provides the preceding-frame reference. Tokens with stronger motion or larger accumulated changes trigger recomputation earlier, while the remaining tokens reuse cached residuals. Only selected tokens are recomputed, with their current K/V entries updated while preserving the causal attention context. Adjusting the cache threshold controls the fraction of updated tokens and the quality–efficiency trade-off.

DisCa. We adapt the learned caching prediction approach of DisCa (Zou et al., 2026) to autoregressive video diffusion. The predictor receives the current noisy latent and the preceding denoising step's final hidden state from the same chunk, without ATC's cross-chunk feature fusion branch. Its DiT blocks incorporate causal KV caching to access historical video context. The predicted final hidden state is decoded into velocity by the frozen output head. We use the same anchor selection and update strategy as ATC, allowing the anchor to come from either a full evaluation or an accepted prediction. For a fair comparison, we control predictor capacity and offline training conditions by using the same number of DiT blocks, training data, regression losses, and main optimization settings as ATC. This comparison evaluates our autoregressive adaptation of DisCa rather than reproducing its original training recipe. Depending on the target speedup, the default schedule is `FFPF` or `FPPF`, replacing one or two full DiT evaluations per subsequent chunk, respectively.

B Additional quantitative and qualitative results

B.1 Additional quantitative comparisons

Tab. 5 compares Self Forcing in the slow setting and Causal Forcing at 81 frames using the VBench subset and efficiency metrics described in Appendix A. The step-reduction baseline is Original 3-step for Self Forcing and Original 2-step for Causal Forcing. For Self Forcing, TGuide uses confidence-aware dynamic scheduling with K08, targeting an average of eight accepted predictions per video; implementation details are provided in Appendix C. It achieves the highest VBench score among accelerated methods and the largest speedup in this setting. For Causal Forcing, TGuide uses fixed `FPPF` and achieves the highest VBench score among accelerated methods at a latency comparable to DisCa's.

Table 5: Additional comparisons on 81-frame videos using the VBench subset. Ours uses dynamic K08 on Self Forcing and fixed `FPPF` on Causal Forcing. Naive step reduction uses three and two steps, respectively.

Method	Self Forcing: Slow				Causal Forcing			
	VBench ↑	TFLOPs ↓	Latency (s) ↓	Speedup ↑	VBench ↑	TFLOPs ↓	Latency (s) ↓	Speedup ↑
Original 4-step	80.68	793	6.53	1.00×	81.58	793	6.48	1.00×
Original 3 / 2-step	80.42	611	4.84	<u>1.35</u>×	81.05	429	3.49	**1.86**×
TeaCache	<u>80.47</u>	611	4.98	1.31×	<u>81.17</u>	493	3.99	1.62×
TaylorSeer	80.40	611	5.06	1.29×	80.60	457	3.76	1.72×
MotionCache	80.35	626	5.04	1.30×	80.85	451	3.75	1.73×
DisCa	<u>80.49</u>	618	4.94	<u>1.32</u>×	<u>81.28</u>	443	3.60	<u>1.80</u>×
Ours	**80.65**	559	4.67	**1.40**×	**81.40**	450	3.63	<u>1.79</u>×

B.2 Full VBench results on Self Forcing

Tab. 6 provides the 16 individual dimension scores and three aggregate scores at 81 and 333 frames for all methods in Tab. 1. All dimension scores are raw values multiplied by 100, before the nor-

malization used for the aggregate scores. DisCa and TGuide are evaluated with the fixed `FPPF` schedule.

Table 6: Per-dimension and aggregate VBench results on 81- and 333-frame Self Forcing generation. Dimension scores are raw values multiplied by 100; Quality, Semantic, and Total Scores are normalized weighted percentages. Higher is better for all metrics. Original denotes the original 4-step model. Bold indicates the best accelerated method in each frame setting; aggregate rankings use scores before final rounding.

81 frames							
Dimension	Original 4-step	Original 2-step	TeaCache	TaylorSeer	MotionCache	DisCa	Ours
Subject Consistency	96.19	96.16	96.15	95.52	95.79	95.80	**96.17**
Background Consistency	96.96	96.90	97.01	95.72	96.88	96.93	**97.22**
Temporal Flickering	98.98	99.12	99.17	99.08	99.08	99.10	**99.19**
Motion Smoothness	98.63	98.58	98.70	**98.72**	98.68	98.58	98.68
Dynamic Degree	58.33	**56.94**	51.39	52.78	54.17	54.17	55.56
Aesthetic Quality	67.20	67.04	66.79	66.79	66.88	66.88	**67.25**
Imaging Quality	68.93	68.59	68.81	**69.44**	68.95	68.63	68.89
Object Class	95.89	**95.73**	94.94	95.02	94.78	94.86	94.15
Multiple Objects	86.74	86.05	85.90	85.37	**86.89**	86.13	86.05
Human Action	97.00	96.00	97.00	96.00	96.00	96.00	**98.00**
Color	88.22	88.41	87.99	85.63	87.48	**88.62**	88.60
Spatial Relationship	84.60	84.34	85.60	**86.12**	85.23	84.26	84.94
Scene	59.08	58.94	**60.32**	59.74	60.10	59.23	59.01
Appearance Style	20.59	20.58	20.62	**20.64**	20.63	20.57	20.58
Temporal Style	26.69	26.59	26.81	**27.04**	26.80	26.71	26.72
Overall Consistency	27.73	27.69	27.84	**28.01**	27.80	27.80	27.77
Quality Score	84.63	84.46	84.14	83.93	84.25	84.17	**84.59**
Semantic Score	82.87	82.59	**83.01**	82.69	82.85	82.62	82.80
Total Score	84.28	84.09	83.91	83.68	83.97	83.86	**84.23**
333 frames							
Dimension	Original 4-step	Original 2-step	TeaCache	TaylorSeer	MotionCache	DisCa	Ours
Subject Consistency	93.13	**93.66**	91.69	88.88	92.13	92.19	93.42
Background Consistency	93.81	94.02	92.67	90.65	92.50	93.14	**94.13**
Temporal Flickering	98.90	**99.11**	99.07	98.94	99.05	99.07	99.09
Motion Smoothness	98.63	98.62	98.61	98.61	**98.67**	98.58	98.50
Dynamic Degree	55.56	52.78	52.78	55.56	52.78	58.33	**59.72**
Aesthetic Quality	64.63	**64.48**	63.35	63.03	62.97	63.29	64.15
Imaging Quality	69.87	**68.90**	68.18	68.45	68.59	68.39	68.70
Object Class	93.59	**93.99**	91.69	90.35	91.85	92.25	92.41
Multiple Objects	83.16	80.72	77.82	78.13	79.19	80.11	**81.40**
Human Action	97.00	96.00	**97.00**	95.00	96.00	96.00	**97.00**
Color	82.80	82.51	80.48	73.16	79.96	81.96	**82.97**
Spatial Relationship	81.91	82.85	80.33	80.07	80.19	82.56	**83.13**
Scene	54.94	54.43	56.47	**58.50**	57.85	57.63	57.05
Appearance Style	21.03	21.07	20.89	**21.27**	20.93	20.86	20.83
Temporal Style	26.96	26.89	27.07	26.85	**27.13**	27.09	26.92
Overall Consistency	27.83	27.86	28.05	**28.13**	27.99	28.08	27.83
Quality Score	82.93	82.76	81.82	81.05	81.89	82.44	**83.12**
Semantic Score	81.04	80.71	80.06	79.26	80.24	80.84	**81.06**
Total Score	82.55	82.35	81.47	80.69	81.56	82.12	**82.71**

B.3 Additional qualitative results

Fig. 5 provides additional Self Forcing comparisons over 20-second sequences, covering people, animals, moving vehicles, and outdoor scenes. At a $1.89\times$ denoising speedup, TGuide preserves recognizable subjects and coherent scene structure in the later frames. For example, the bear retains detailed fur and a clear body outline, while the forest path remains well defined as the viewpoint advances. Several competing methods exhibit increasing texture artifacts, loss of detail, or appearance drift by 20 seconds, most visibly in the workshop, bear, and nighttime street examples. These observations complement the quantitative results at 333 frames and support the benefit of historical trajectory guidance for maintaining visual quality over extended generation.



Figure 5: Additional qualitative comparisons on Self Forcing. Each example shows frames at 0, 10, and 20 seconds under matched generation conditions, with denoising speedups indicated for each method.

C CONFIDENCE-AWARE DYNAMIC SCHEDULING

Confidence-head fitting. After predictor training, we freeze the generator and ATC and collect student rollouts using the deployment sampling schedule. We save the visited states for offline confidence-head training. At each collected state, the frozen full DiT provides a reference hidden state under the same noisy latent, timestep, and historical conditions as the ATC prediction. The confidence head is trained from scratch to estimate the resulting local prediction error. The head combines mean-pooled predictor features with confidence-token attention pooling, conditioned on chunk position, denoising step, and hidden-state statistics. The head predicts log relative hidden-state error:

$$\epsilon = \frac{\|\hat{h} - h^F\|_2}{\|h^F\|_2 + \delta}, \qquad \hat{y} = \mathrm{Conf}(z), \qquad \hat{\epsilon} = \exp(\hat{y}), \tag{9}$$

where z collects the head inputs, $\hat{h}$ and h^F are the ATC prediction and full-DiT reference evaluated at the same collected rollout state, and $\delta > 0$ ensures numerical stability. We train the head with the SmoothL1 objective:

$$\mathcal{L}_{\mathrm{conf}} = \mathrm{SmoothL1}(\hat{y}, \log(\epsilon + \delta)) \,. \tag{10}$$

Training and validation are split by prompt, and we select the checkpoint with the lowest validation loss. For Self Forcing, training uses a global batch size of 256 for up to 20 epochs, with a peak learning rate of 10^{-4}.

Risk estimation. At inference, we freeze ATC and the confidence head and select the acceptance threshold τ separately. At each predictor-eligible step i in chunk j, ATC first produces a candidate final hidden state $\hat{h}_{j,i}$ from the current noisy latent and cached history. The confidence head then estimates its log relative error $\hat{y}_{j,i}$ without evaluating the full DiT. We convert this estimate into a scheduling risk:

$$\hat{\epsilon}_{j,i} = \exp(\hat{y}_{j,i}), \qquad r_{j,i} = \hat{\epsilon}_{j,i} \frac{|t_i - t_{i+1}|}{T}(1 + \beta\alpha_j), \qquad \alpha_j = \frac{K - 1 - j}{K - 1 - j_0}, \tag{11}$$

where K is the total number of chunks, j_0 is the first predictor-eligible chunk, and t_N is the terminal timestep after the last denoising evaluation. We use $T = 1000$ and $\beta = 2$. The outgoing denoising span accounts for the sampling update affected by the prediction, while α_j decreases from one at j_0 to zero at the final chunk to assign greater weight to errors that can propagate through more subsequent chunks.

Acceptance and full-DiT fallback. If $r_{j,i} \le \tau$, we accept the candidate hidden state. Otherwise, we discard the candidate and evaluate the full DiT on the same noisy latent, timestep, and conditions before making any sampling update:

$$h_{j,i} = \begin{cases} \hat{h}_{j,i}, & r_{j,i} \le \tau, \\ \mathrm{DiT}_\theta^{\mathrm{final}}(x_i^j, t_i, \mathcal{C}^j), & r_{j,i} > \tau, \end{cases} \tag{12}$$

where $\mathrm{DiT}_\theta^{\mathrm{final}}$ denotes the final hidden state produced by the full DiT, and $\mathcal{C}^j$ contains the external conditions and historical KV context. Thus, the full DiT is evaluated only when the candidate is rejected at a predictor-eligible step. Steps that are not predictor-eligible directly use the full DiT.

Sampling and cache updates. The selected hidden state is decoded through the frozen output head and used by the base model's sampler:

$$v_{j,i} = \mathrm{Head}(h_{j,i}, t_i), \qquad x_{i+1}^j = \mathcal{S}_i\Big(x_i^j, v_{j,i}\Big) \,. \tag{13}$$

We also cache $h_{j,i}$ for subsequent predictions, including same-chunk anchoring and historical guidance at the matching timestep of the next chunk. After a fallback, this cache therefore contains the full-DiT hidden state rather than the rejected prediction. Both accepted and fallback outputs can consequently become references for later ATC calls. Because ATC and the confidence head run before the acceptance decision, rejected attempts incur their computation in addition to the full-DiT evaluation; all reported efficiency measurements include this cost.

Scheduling ablation. Tab. 7 compares fixed schedules and confidence aware dynamic scheduling on 81-frame Self Forcing videos. K06, K08, and K10 denote target average counts of 6, 8, and 10 accepted predictions per video, respectively. The dynamic configurations use the same predictor and confidence head, varying only the acceptance threshold. K08 improves generation quality over both `FPFF` and `FFPF` while reducing latency, whereas K10 provides a faster operating point with a modest quality tradeoff. Compared with fixed `FPPF` under the same VBench subset evaluation, K08 provides higher generation quality at a lower acceleration level. This ablation complements the comparisons with other acceleration methods in Appendix B.1.

Table 7: Dynamic schedule ablation on 81-frame Self Forcing videos, evaluated on the VBench subset.

Method	VBench ↑	TFLOPs ↓	Latency (s) ↓	Speedup ↑
Original 4 steps	80.68	793	6.53	1.00×
`FPFF`	80.55	622	4.99	1.31×
`FFPF`	80.57	622	4.99	1.31×
Dynamic K06	<u>80.62</u>	613	5.10	1.28×
Dynamic K08	**80.65**	559	4.67	<u>1.40</u>×
Dynamic K10	<u>80.58</u>	507	4.24	<u>1.54</u>×
`FPPF`	80.55	452	3.62	**1.80**×

D Training details

D.1 Stage 1: offline feature regression

Shared supervision. Each predictor block uses the historical and text attention K/V from its corresponding source layer. Supervision targets the final hidden states of the complete teacher DiT and their corresponding velocities, rather than the intermediate outputs of the selected layers. Both the within-chunk anchor and the previous chunk's same-timestep reference are taken from offline teacher trajectories. The denoising residual output and cross-chunk correction output are zero-initialized, so the prediction initially copies the anchor. The first chunk provides context only, while each subsequent chunk contributes three prediction positions. We optimize a combination of hidden-state and velocity MSE:

$$\mathcal{L}_{\mathrm{S1}} = \lambda_h \,\mathrm{MSE}(\hat{h}, h^F) + \lambda_v \,\mathrm{MSE}(\mathrm{Head}(\hat{h}, t_i), v^F). \tag{14}$$

Here h^F and v^F are the final hidden state and velocity from the full DiT evaluated at the same input state. We set $\lambda_h = 0.1$ and $\lambda_v = 1$, without additional timestep weighting or relative-error normalization.

Self Forcing and Causal Forcing. ATC uses a single predictor block initialized from layer 17 of the corresponding pretrained generator. The source-layer selection study for Self Forcing is provided in Appendix F. We collect 81-frame teacher rollouts for offline supervision. Training cycles through shuffled chunk–timestep groups; each update samples 64 distinct prompts from one group, keeping the historical context length consistent within a batch. We train for 2000 updates with a global batch size of 64 using AdamW, with betas of $(0.9, 0.95)$, weight decay of 0.01, and gradient clipping at 1.0. The learning rates are 10^{-4} for the input/fusion parameter group and 10^{-5} for the predictor block. We use cosine learning-rate decay. The final checkpoint initializes Stage 2.

HY-WorldPlay. ATC uses two predictor blocks initialized from layers 1 and 52 of the pretrained generator, with zero-based indexing. Offline supervision uses 125-frame teacher rollouts conditioned on the initial image, text prompt, and camera trajectory, with the shared supervision protocol and loss defined above. We train for 2000 updates with an effective batch size of 64, using AdamW with betas $(0.9, 0.95)$, weight decay 0.01, and gradient clipping at 1.0. The learning rates are 10^{-4} for the fusion module and residual outputs and 10^{-5} for the two predictor blocks, with cosine decay.

D.2 Stage 2: self-rollout on-policy distillation

Student rollout and random exit. Self Forcing and Causal Forcing use the same Stage 2 training procedure, initialized from the final Stage 1 checkpoint. Each rollout generates an 81-frame video

with the base generator frozen. For each predictor update, we randomly choose an exit step $i_{\text{exit}} \in \{1, 2, 3\}$, shared across chunks. The first chunk uses full DiT evaluations up to the selected exit. For subsequent chunks, the first step uses the full DiT, and ATC drives the remaining sampling steps and updates the feature cache. Each chunk exits with its predicted clean latent, which provides context for the next chunk. Varying the exit exposes ATC to different prediction depths and to historical features produced by its own earlier predictions. Gradients are retained only for the exit-step ATC predictions; earlier rollout states, anchor features, historical features, and KV caches are detached.

DMD supervision. Following Self Forcing (Huang et al., 2025), we use a frozen non-causal Wan2.1-T2V-14B real-score teacher and a trainable non-causal Wan2.1-T2V-1.3B fake-score critic for DMD (Yin et al., 2024b;a) supervision of the generated video. The score networks evaluate the complete video, while the ATC objective excludes the first chunk and adds no Stage 1 reconstruction loss. The critic is trained with flow matching on detached student samples, including all chunks.

Training parameters. We perform 2000 predictor updates with a global batch size of 8, with five critic updates on fresh samples before each predictor update. AdamW uses betas $(0, 0.999)$, weight decay 0.01, and learning rates 10^{-5}, 10^{-6}, and 10^{-5} for the ATC fusion module, predictor block, and critic, respectively. We use cosine learning-rate decay. Predictor EMA starts at update 200 with decay 0.99. The DMD score timestep is shared across the video, sampled from integer timesteps in $[0, 1000)$, warped with shift 5, and clipped to $[20, 980]$. Teacher guidance uses $D_{\text{cond}} + 3(D_{\text{cond}} - D_{\text{uncond}})$.

HY-WorldPlay. For training efficiency, we fine-tune HY-WorldPlay's predictor using direct teacher supervision on student rollouts, avoiding the additional score-network training and alternating updates required by DMD. We initialize ATC from Stage 1 and keep the full teacher frozen. Each training rollout generates 125 frames. The first chunk uses the full DiT throughout; subsequent chunks use the full DiT at the first denoising step and ATC at the remaining steps. Predicted velocities advance the noisy latents, and the resulting hidden states and generated chunks provide the inputs and historical context for later predictions.

At each predictor step, the full teacher evaluates the same noisy latent reached by the student and supplies the target final hidden state h^F and velocity v^F. We minimize

$$\mathcal{L}_{\text{HY}} = \mathbb{E}\Big[\lambda_h \ \text{MSE}(\hat{h}, h^F) + \lambda_v \ \text{MSE}(\hat{v}, v^F)\Big]. \tag{15}$$

The expectation averages over predictor steps on student rollouts, using the same loss weights as Stage 1. The teacher outputs serve only as supervision; the student outputs drive subsequent sampling updates. Rollout states and cached features are detached between denoising steps and chunks, so each prediction receives local supervision without backpropagation through the complete trajectory.

We train for 2000 updates with an effective batch size of eight trajectories, using learning rates of 5×10^{-5} for the fusion module and 5×10^{-6} for the predictor block. Weight decay and gradient clipping are set to 0.01 and 1.0, respectively.

D.3 Dataset

Self Forcing and Causal Forcing. We sample training prompts from VidProM (Wang & Yang, 2024). Stage 1 uses offline trajectories generated by the corresponding full model, with final hidden states and velocities providing supervision for the predictor. Stage 2 samples prompts from the filtered, extended VidProM pool and generates training trajectories online with the current predictor.

HY-WorldPlay. Both training stages use randomly selected prompts and their corresponding initial images from the VBench I2V set. The image–prompt pairs used for training are disjoint from those used for evaluation, with no overlap in either input images or prompts. We combine these image–prompt pairs with camera trajectories that move forward then backward, move left then right, tilt up then down, or pan left then right. Stage 1 uses offline full-teacher trajectories, while Stage 2 generates student rollouts under the same image, prompt, and camera conditions and obtains teacher supervision at the visited states.

Table 8: Validation velocity MSE across all source layers.

Source layer	Val. MSE ↓	Source layer	Val. MSE ↓	Source layer	Val. MSE ↓
0	0.052675	10	0.049378	20	0.047325
1	0.051242	11	0.048483	21	0.046994
2	0.048375	12	0.049627	22	0.058777
3	0.048407	13	0.048251	23	0.048755
4	0.048115	14	0.049588	24	0.047499
5	0.048193	15	0.049718	25	0.048287
6	0.047933	16	0.047317	26	0.048326
7	0.049241	17	**0.046459**	27	0.049492
8	0.048467	18	0.047587	28	0.051012
9	0.049020	19	0.047851	29	0.050500

Table 9: Initialization controls with layer-17 K/V fixed, evaluated by validation velocity MSE.

Initialization	Val. MSE ↓
Complete teacher block	**0.046459**
Teacher + zeroed branch outputs	0.055906
Xavier + zeroed branch outputs	0.057737
Xavier	0.057085
All-zero block	0.060460

E Details of the Motivating Experiments

Correlation along denoising time and video time. Fig. 2(a) measures token cosine similarity of final hidden states between consecutive denoising timesteps and between the previous chunk's last temporal slot and the current chunk at the same timestep. The 50-step references use a bidirectional DiT teacher for Self Forcing and a causal DiT teacher for Causal Forcing. Spatial alignment uses optical flow from generated RGB frames; hatched bars show the aligned results. Alignment increases the 4-step cross-chunk mean by $0.0138/0.0164$ for Self Forcing/Causal Forcing. Optical-flow alignment is used only for this diagnostic and is not part of ATC inference.

Global impact of local approximation. Fig. 2(b) applies `FRRF` to one chunk at a time in a 165-frame rollout, keeping every other chunk `FFFF`. Each R reuses that chunk's first full-step velocity. Curves average full-video PSNR, SSIM, and LPIPS over the first 10 MovieGenVideoBench-extended prompts (seed 0), relative to matched `FFFF` references. These full-video metrics assess the overall impact of approximation placement, including the different lengths of the remaining rollout.

F Predictor block selection on Self Forcing

We study source-layer selection and initialization using a single-block predictor. Candidates are trained under the same offline regression protocol and compared by validation velocity MSE on held-out teacher states. Each source layer supplies both the initial block weights and the corresponding attention K/V; layer indices are zero-based.

F.1 Source layer and initialization

Tab. 8 reports all 30 source layers. Layer 17 has the lowest validation velocity MSE (0.046459) among the single-block candidates. These results motivate using layer 17 for subsequent Self Forcing predictor training.

Four additional controls keep layer-17 K/V fixed and change only block initialization (Tab. 9). Initializing from the complete teacher block yields the lowest validation velocity MSE among the tested initialization schemes under the same training budget.