Trajectory Forcing: Exploiting Diffusion Trajectories for Autoregressive Long Video Generation

Bohan Wang* **Shuo Chen*** **Yunzhi Li***
Zhaozheng Chen **Junzhe Zhang** **Hanwang Zhang†**
Huawei Noah Research Institute

Project Page: https://wangbohan97.github.io/Trajectory-Forcing/

Abstract

Autoregressive (AR) modeling is arguably the only way to extend a fixed-length video diffusion model to longer or even streaming videos. However, existing AR diffusion methods use only the previously predicted chunk as context for the next, discarding the intermediate predictions that constitute the chunk's generation history. In contrast, conventional AR models such as LLMs condition on the entire history of preceding tokens. To this end, we propose **Trajectory Forcing (TraF)**, an AR diffusion method that exploits the denoising trajectories of previous video chunks as context. TraF operates under two training regimes: (1) with ground-truth videos, where pseudo-trajectories are constructed from Gaussian interpolations and re-predicted by the model, and (2) with a teacher model, where real trajectories are obtained from the student's own denoising rollout. The first regime naturally provides a strong initialization for the second. On the 5-second VBench benchmark, TraF achieves a Total score of 85.15, outperforming Self Forcing (+0.84) and Causal Forcing (+1.11), with Quality and Semantic improving simultaneously. On 30-second generation—6× the training horizon—TraF achieves the highest overall quality among all compared methods.

1 Introduction

Recent video diffusion models have made rapid progress in generating realistic short videos with complex motion [15, 21, 5]. However, most models still operate within a fixed-length temporal window, limiting both duration and streaming capability. Thus, Autoregressive Diffusion (AR Diffusion), which treats fixed-length diffusion generation as tokens in next-token prediction, is arguably the only approach for jointly modeling temporal sequences of diffusion-generated windows, enabling variable-length generation at inference time for applications such as interactive content creation [30] and world simulation [16].

AR models a temporal sequence of length N via a causal Bayesian factorization:

$$p(x_1, \ldots, x_N) \;=\; p(x_1) \prod_{i=2}^{N} p\big(x_i \mid x_1, \ldots, x_{i-1}\big). \tag{1}$$

In language models, x_i denotes an atomic token that does not require further generation (Figure 1a). However, in AR diffusion, the community often assumes that x_i corresponds to a video chunk, overlooking that x_i is no longer "atomic" in video generation. Instead, x_i should consist of T denoising steps, *i.e.*, $x_i \coloneqq (x_{(i,1)}, \ldots, x_{(i,T)})$, where $x_{(i,1)}$ is standard Gaussian noise and $x_{(i,T)}$ is

*Equal contribution. †Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

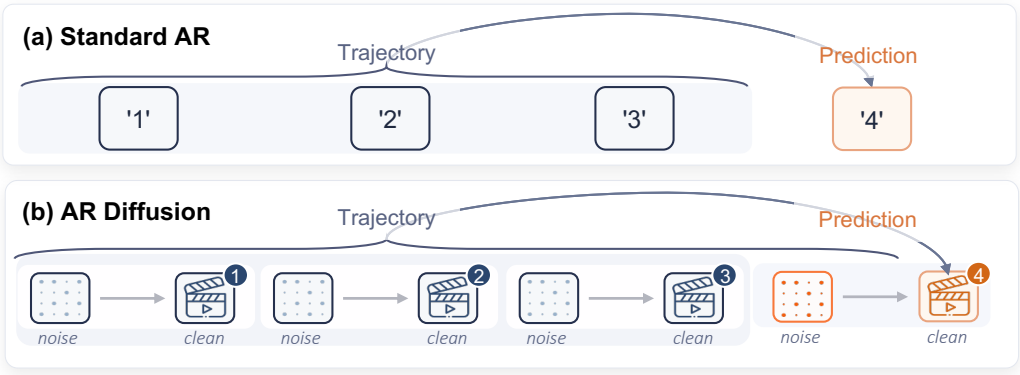


Figure 1: **Standard AR vs. AR diffusion.** (a) In standard AR (Eq. 1), each token is atomic and requires no further generation. The next token is naturally conditioned on all preceding tokens. (b) In AR diffusion, each chunk is generated through T denoising steps that form a trajectory. Following Eq. (2), each denoising step should be conditioned on the full trajectories of all preceding chunks.

the "clean" video. Therefore, a correct formulation of AR diffusion should be:

$$\begin{aligned} p(x_1, \ldots, x_N) &= p\big(x_{(1,1)}, \ldots, x_{(1,T)},\ \ldots,\ x_{(N,1)}, \ldots, x_{(N,T)}\big) \\ &= p(x_{(1,1)}) \prod_{(i,t)=(1,2)}^{(N,T)} p\big(x_{(i,t)} \mid x_{(1,1)}, \ldots, x_{(i,t-1)}\big). \end{aligned} \tag{2}$$

After this expansion, each element $x_{(i,t)}$ is atomic, as it corresponds to a single denoising step and requires no further internal generation. Thanks to Eq. (2), we can immediately know that existing methods do not consider the *full trajectory* of each x_i. For example, in Teacher Forcing [11], the conditioning on preceding chunks is reduced to the set of endpoints $\{x_{(n,T)}\}_{n<i}$, discarding all the intermediate points $\{x_{(n,t)}\}_{n<i,\, t<T}$ along the trajectory (Figure 1b). A detailed comparison is presented in Table 1.

After properly formulating AR diffusion in Eq. (2), we must address error accumulation due to train–test mismatch. A natural approach is to apply Reinforcement Learning (RL, or self-rollout forcing) on top of Supervised Learning (SL, or teacher forcing). In the SL stage, the model in Eq. (2) is trained using ground-truth videos. During inference, however, ground truth is unavailable, and the model must condition on its own imperfect predictions. This mismatch leads to error accumulation along the autoregressive sequence. In contrast, the RL stage trains the model on its own autoregressive rollouts, with supervision provided only by a trajectory-level reward model, thereby better aligning training and inference conditions. From this perspective, Self Forcing [9] can be viewed as a lightweight instance of RL, as it trains on autoregressive rollouts and uses a teacher model to reward the generated videos. We discuss related work for both stages in Section 2.

In this paper, we propose **Trajectory Forcing (TraF)**, which builds on the trajectory-level AR diffusion formulation in Eq. (2) for both the SL and RL stages. Directly conditioning on the full trajectory would multiply the KV-cache size per chunk by the number of denoising steps, significantly increasing memory usage and attention cost, and rendering training impractical on current hardware. Instead, we condition on a single trajectory point per chunk selected at each denoising step (§3.2), replacing the static endpoint used by existing methods.

- **SL stage** (§3.1). In the SL stage, the model is trained with flow matching [14] on ground-truth videos. Since ground-truth videos do not provide model-generated denoising trajectories, we construct a pseudo-trajectory via linear interpolation. Because this path is a straight line, the entire trajectory can be analytically computed from a *single* velocity prediction (Algorithm 1). We show that TraF achieves a total score of 84.39, outperforming Teacher Forcing (**+0.92**) and Diffusion Forcing (**+0.32**) (Table 4).
- **RL stage** (§3.2). In the RL stage, the model is trained on its own autoregressive rollouts under reward-model supervision via distribution matching distillation [32]. During rollout, the T-step

	Method	SL	RL
Fixed-point	Teacher Forcing [26, 11, 22, 34]	GT $x_{(n,T)}$	✗
	Diffusion Forcing [1, 20, 33, 27, 6, 17]	Corrupted $x_{(n,T)}$	✗
	Self Forcing [9, 36, 13, 29, 3]	✗	Endpoint $x_{(n,T)}$
Trajectory	**Trajectory Forcing (Ours)**	**Trajectory** $\boldsymbol{x}_{(n,\cdot)}$	**Trajectory** $\boldsymbol{x}_{(n,\cdot)}$

Table 1: **AR diffusion methods categorized by their use of the denoising trajectory.** Each column reports the conditioning used under the corresponding training stage. ✗ indicates that the method does not use that stage. Prior methods condition on a fixed point regardless of the denoising step, whereas Trajectory Forcing conditions on the denoising trajectory $x_{(n,\cdot)}$.

denoising process of each chunk naturally produces a trajectory, from which a trajectory point is selected at each denoising step (Algorithm 2). TraF achieves a total score of **85.15**, outperforming Self Forcing by **0.84** and Causal Forcing by **1.11** (Table 2). For 30-second generation, it achieves the highest total score (**73.98**) among methods without long-video-specific training (Table 3).

2 Related Work

We organize related AR video diffusion methods by their training stage and use of the denoising trajectory (Table 1).

Supervised learning. Teacher Forcing [26] is the standard training stage in current AR video diffusion systems [11, 22, 34], where each chunk is conditioned on the ground-truth endpoints $\{x_{(n,T)}\}_{n<i}$ of preceding chunks. Diffusion Forcing [1] and related methods [20, 33, 27, 6, 2] replace these clean endpoints with corrupted ground-truth versions, partially mitigating the train–test mismatch. In both cases, the model still conditions on a single point derived from ground truth, and the denoising trajectory $\{x_{(n,\cdot)}\}_{n<i}$ remains unused.

Reinforcement learning. Self Forcing [9] trains on the model's own autoregressive rollouts, conditioning each chunk on the endpoint predictions $\{x_{(n,T)}\}_{n<i}$ of preceding chunks. This removes the train–test mismatch but still conditions on a single trajectory point. Causal Forcing [36] extends this stage with causal ODE initialization and distribution-matching distillation. Other methods explore rolling diffusion windows [13], minute-scale generation [29, 3, 35], and diagonal denoising paths [12]. The concurrent HiAR [37] takes a different route by jointly denoising all chunks in a step-first order, thereby departing from the sequential AR formulation in Eq. (2). Artifacts such as color shift and blurring have been observed even when the train–test mismatch is removed [34, 37], indicating that the mismatch is not the only source of quality degradation. As Eq. (2) suggests, each denoising step should be conditioned on the trajectories of preceding chunks, yet existing methods still condition on a fixed point regardless of the denoising step.

3 Method

Trajectory Forcing is motivated by the trajectory-level formulation in Eq. (2), which factorizes autoregressive video diffusion at the level of individual denoising steps: each $x_{(i,t)}$ is conditioned on all preceding trajectory points $x_{(1,1)}, \ldots, x_{(i,t-1)}$. Existing methods reduce this conditioning to the set of clean endpoints $\{x_{(n,T)}\}_{n<i}$, discarding all intermediate points along the trajectory. Directly conditioning on the full trajectory, however, would multiply the representations per chunk by the number of denoising steps, significantly increasing attention computation and memory usage and making training impractical on current hardware. Instead, we condition on a single trajectory point per chunk selected at each denoising step, replacing the static endpoint used by existing methods. We implement this trajectory-level conditioning in two stages. Section 3.1 constructs pseudo-trajectories for supervised training on ground-truth videos, and Section 3.2 leverages the real trajectories that emerge from autoregressive rollouts.

Algorithm 1 SL: Training on Ground Truth

Require: Dataset $\mathcal{D}$, model v_θ

loop
  Sample $(x_0^{1:N}, \text{prompt})$ from $\mathcal{D}$
  Sample $t \in [0, 1]$
  for $i = 1, \ldots, N$ **do** ▷ pseudo-traj. (no grad)
    $\epsilon_i \sim \mathcal{N}(0, I)$
    Sample $\sigma_i \in [0, 1]$
    $z_i \leftarrow (1-\sigma_i)\, x_0^i + \sigma_i\, \epsilon_i$
    $x_{(i)} \leftarrow z_i - \sigma_i \cdot v_\theta(z_i, \sigma_i, x_{(<i)})$
  end for
  Sample $\epsilon_i' \sim \mathcal{N}(0, I)$ for each i
  $x_t^i \leftarrow (1-t)\, x_0^i + t\, \epsilon_i'$
  $v_i^* \leftarrow \epsilon_i' - x_0^i$
  $\mathcal{L} \leftarrow \frac{1}{N} \sum_i \left\| v_\theta(x_t^i,\, t,\, x_{(<i)}) - v_i^* \right\|^2$
  Update θ
end loop

Algorithm 2 RL: Training on Self-Rollouts

Require: Schedule $\{\sigma_1, \ldots, \sigma_T\}$, student G_ϕ, γ

loop
  Sample prompt; $s \sim \text{Uniform}\{1, \ldots, T\}$
  for $i = 1, \ldots, N$ **do** ▷ real traj. (rollout)
    $x_{\sigma_1} \sim \mathcal{N}(0, I)$
    for $t = 1, \ldots, T$ **do**
      $k^* \leftarrow S_\gamma(p_t)$ ▷ Eq. 4
      if $t{=}s$**:** enable gradient
      $\hat{x}_0 \leftarrow G_\phi(x_{\sigma_t},\, \sigma_t,\, x_{(<i,\, k^*)})$
      $x_{(i,t)} \leftarrow \hat{x}_0$
      if $t{=}s$**:** disable gradient
      if $t{<}T$**:** ODE step to σ_{t+1}
    end for
  end for
  Compute $\mathcal{L}_{\text{DMD}}$; update ϕ
end loop

3.1 SL Stage: Training on Ground Truth

The model is trained with flow matching [14] on ground-truth videos, with x_0^i denoting chunk i. Eq. (2) requires conditioning on the denoising trajectory of each preceding chunk, $x_{(n,1)}, \ldots, x_{(n,T)}$, namely the clean-signal estimates produced at successive denoising steps. However, ground-truth videos are not produced by denoising, so no such trajectory exists. We address this by constructing a *pseudo-trajectory* via linear interpolation. The flow matching forward process $x_t = (1-t)\, x_0 + t\, \epsilon$ interpolates linearly between data and noise. Along this straight line, the oracle velocity $v^* = \epsilon - x_0$ is constant and recovers the same x_0 at every noise level. With the learned predictor, each sampled pseudo-state is obtained from a *single* velocity prediction, and estimates at different noise levels need not coincide. These estimates constitute the entire pseudo-trajectory (Algorithm 1). The trajectory conditioning prescribed by Eq. (2) then takes the following form:

$$p\big(x_{(i,t)} \mid \underbrace{x_{(1,1)}, \ldots, x_{(1,T)}}_{\text{pseudo-traj.}}, \ \ldots, \ \underbrace{x_{(i-1,1)}, \ldots, x_{(i-1,T)}}_{\text{pseudo-traj.}}\big), \qquad x_{(n,t)} \coloneqq x_t - t \cdot v_\theta, \tag{3}$$

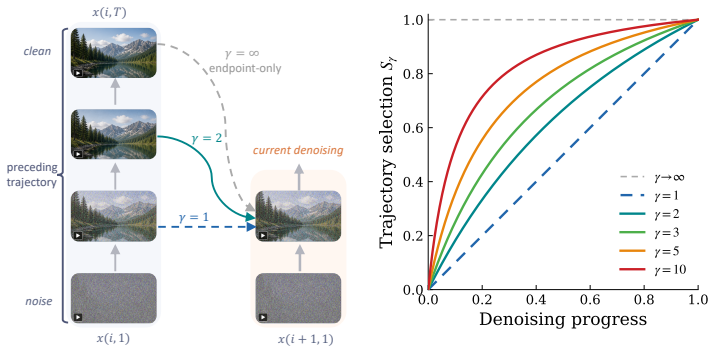
where $x_{(n,t)}$ is the clean-signal estimate at noise level t, obtained from a single velocity prediction.

This is distinct from Diffusion Forcing [1], which conditions directly on a noise-corrupted version of the ground truth, namely a point on the *forward* noising path rather than a clean-signal estimate on the denoising trajectory prescribed by Eq. (2). During training, the noise level is sampled uniformly at each iteration, so the model encounters trajectory points spanning the full fidelity spectrum, from coarse scene layout at high noise to fine texture at low noise [23]. Although the pseudo-trajectory is approximate—all points derive from a single velocity prediction rather than independent denoising steps—the flow matching loss provides a precise learning signal by regressing directly to the ground-truth velocity. The RL stage then replaces this approximation with real trajectories from the model's own rollout (§3.2).

3.2 RL Stage: Training on Self-Rollouts

The model is trained on its own autoregressive rollouts via a student–teacher framework [9]. In this framework, the student G_ϕ generates chunks autoregressively, and Distribution Matching Distillation (DMD) [32] provides the training signal using a frozen teacher score function and a learned critic (Algorithm 2).

Unlike the SL stage, no trajectory construction is needed. During rollout, each chunk undergoes T denoising steps, and each step produces a clean-signal estimate $x_{(i,t)}$ ($t = 1, \ldots, T$). These estimates form the real denoising trajectory $x_{(i,1)}, \ldots, x_{(i,T)}$, which is precisely the object required by Eq. (2). However, conditioning each step of the current chunk on all T trajectory points from every preceding chunk requires attending to T times as many representations. The resulting increase in attention computation and memory usage renders training impractical on current hardware. We



γ	Quality↑	Semantic↑	Total↑
∞	85.04	81.07	84.24
10	85.40	81.50	84.62
5	85.47	81.03	84.58
3	**85.91**	81.97	85.13
2	85.89	**82.20**	**85.15**
1	85.49	82.10	84.82

Figure 2: **Trajectory conditioning and γ ablation.** *Left:* Each denoising step of the current chunk conditions on the preceding chunk's trajectory. γ determines the selected position ($\gamma\to\infty$: endpoint-only conditioning, as in existing methods). *Middle:* Unregularized continuous targets S_γ; finite-grid selection is detailed in Appendix A. *Right:* The tested finite values of γ outperform endpoint-only conditioning in Total. γ=2–3 forms a broad optimum.

therefore select a *single* trajectory point per preceding chunk at each denoising step, replacing the static endpoint $x_{(n,T)}$ used by existing methods. Because different steps select different positions along the trajectory, selected trajectory points provide context across steps without the cost of attending to all points at every step.

Selection function. The selection should reflect the coarse-to-fine structure of denoising [23]. An early step, which is still resolving scene layout, benefits from a coarser trajectory point, while a later step, which is refining textures, benefits from a more refined one. We formalize this intuition using a Wiener-filter-inspired SNR criterion (Appendix A). Let $p = 1 - \sigma \in [0, 1]$ denote the denoising progress, where $p = 0$ corresponds to pure noise and $p = 1$ corresponds to the clean signal. Requiring the selected trajectory point to lead the current step by a factor of γ in amplitude SNR yields a closed-form selection function:

$$S_\gamma(p) = \frac{\gamma\, p}{1 + (\gamma - 1)\, p}, \qquad \gamma \geq 1. \tag{4}$$

At each step with progress p, the model conditions on $x_{(\cdot,\, k^*)}$, selected using the discrete rule in Appendix A. Eq. (4) defines the continuous target; in the discrete conditioning expressions and Algorithm 2, S_γ is shorthand for the regularized nearest-point selection specified in Appendix A. Motivated by Eq. (2), we use the following trajectory-point conditioning:

$$p\big(x_{(i,t)} \mid \underbrace{x_{(1,k^*)},\ \ldots,\ x_{(i-1,k^*)}}_{\text{real trajectory}}\big), \qquad k^* \leftarrow S_\gamma(p_t), \tag{5}$$

where k^* is the selected trajectory-point index. This parameterization includes prior methods as special cases (Figure 2). Specifically, γ=1 gives step-aligned selection, while $\gamma \to \infty$ recovers endpoint-only conditioning (Table 1). Empirically, all tested finite γ outperform the endpoint baseline in Total, with $\gamma \in [2, 3]$ forming a broad optimum (§4.4). All other training components, including gradient truncation to a single random step and KV detachment across chunks, follow standard practice [9].

The two stages share the same goal of implementing the trajectory-level conditioning prescribed by Eq. (2). They differ in trajectory source and supervision. The SL stage constructs a pseudo-trajectory from ground truth and trains with flow-matching regression, while the RL stage operates on real trajectories produced by rollout and trains with DMD. This makes the two stages complementary: the SL stage builds generation capability from precise ground-truth supervision despite approximate trajectories, while the RL stage aligns this capability to real inference trajectories via teacher-model supervision.

Table 2: **Short-video evaluation on VBench** (946 prompts, all 16 dimensions). Result sources are detailed in Appendix C. Throughput and latency are reported in FPS and seconds, respectively. —: not reported.

Method	#Params	Teacher	Throughput (FPS)↑	Latency (s)↓	Total↑	Quality↑	Semantic↑
Bidirectional Video Diffusion							
LTX-Video [7]	1.9B	✗	8.98	13.5	79.83	81.88	71.62
Wan2.1-1.3B [21]	1.3B	✗	0.78	103	83.37	84.30	79.65
Autoregressive Video Diffusion							
MAGI-1 [22]	4.5B	✗	0.19	282	78.88	81.67	67.72
NOVA [4]	0.6B	✗	0.88	4.1	80.31	80.66	78.92
Pyramid Flow [11]	2B	✗	6.70	2.5	80.75	83.41	70.11
SkyReels-V2 [2]	1.3B	✗	0.49	112	81.97	83.96	74.01
Distilled Autoregressive Video Diffusion							
CausVid [33]	1.3B	Wan-14B	17.0	0.69	81.33	83.98	70.72
Rolling Forcing [13]	1.3B	Wan-14B	15.8	0.76	83.07	83.61	80.89
LongLive [29]	1.3B	Wan-14B	18.0	0.65	83.09	83.69	80.69
HiAR [37]	1.3B	Wan-14B	—	—	83.27	84.22	79.46
Causal Forcing [36]	1.3B	Wan-14B	17.0	0.69	84.04	84.59	81.84
Self Forcing [9]	1.3B	Wan-14B	17.0	0.69	84.31	85.07	81.28
Trajectory Forcing (Ours)	1.3B	Wan-14B	15.0	0.78	**85.15**	**85.89**	**82.20**

4 Experiments

4.1 Experimental Setup

Implementation. We build on Wan2.1-T2V-1.3B [21] with causal attention as the base model, generating 832×480 videos at 16 fps. In the SL stage (§3.1), this model is trained with flow matching on ground-truth videos. In the RL stage (§3.2), Wan2.1-T2V-14B serves as the real score network for distribution matching distillation (DMD) [32]. In the RL stage, we use the same backbone architecture and teacher as Self Forcing [9] and Causal Forcing [36]. TraF is initialized from its SL checkpoint, while the baselines retain their respective original initializations. Specifically, we use VidProM [24] prompts rewritten by Qwen2.5-7B-Instruct [19], a batch size of 64, and the Adam optimizer (lr$=2\times10^{-6}$, $\beta_1=0$, $\beta_2=0.999$). Full training details are provided in Appendix B.

Evaluation. For short-video evaluation, we adopt the standard VBench [10] protocol with 946 prompts across all 16 dimensions. Scores are aggregated into Quality and Semantic following the official VBench weighting, with Total $=(4\times\text{Quality}+\text{Semantic})/5$ on a 0–100 scale. For long-video evaluation, we use 128 prompts from MovieGenBench [18] to generate 30-second videos, with 6 segments of 5 s each. Following CausVid [33], we group the 9 evaluated VBench dimensions into *Temporal Quality* (subject consistency, background consistency, temporal flickering, motion smoothness, dynamic degree), *Frame Quality* (aesthetic quality, imaging quality), and *Text Alignment* (temporal style, overall consistency), with Total $=$ (Temporal Quality $+$ Frame Quality $+$ Text Alignment) $/\,3$. We also include Quality Drift [13], defined as the absolute difference in imaging quality between the first and last 5 seconds of each video, to reflect the severity of error accumulation. We also report the text-alignment sub-score from VideoScore2 [8]. For fair comparison, we equip all block-first methods (SF, CF, and Ours) with the same sliding-window inference and attention sink [13, 28]. The first chunk's KV cache is permanently retained as global context, and subsequent chunks attend to a local window of 21 latent frames. HiAR [37], Rolling Forcing [13], and LongLive [29] use their respective inference architectures. Evaluation details are provided in Appendix C.

4.2 Short-Video Results

Table 2 reports VBench scores for short-video generation. Trajectory Forcing achieves the highest Total (85.15), Quality (85.89), and Semantic (82.20) scores among all compared methods. Compared with the reported results, it outperforms Self Forcing by +0.84 and Causal Forcing by +1.11 in Total.

Table 3: **Long-video evaluation** (128 prompts, 30 s). Temporal Quality, Frame Quality, and Text Alignment are axis aggregates following CausVid [33]. Total is their equal-weighted average. The VideoScore2 [8] column reports the text-alignment sub-score. Gray: long-video-specific training.

Method	Temporal Quality ↑	Frame Quality ↑	Text Align. ↑	Total ↑	Video-Score2 ↑	Quality Drift ↓
CausVid [33]	83.97	62.66	58.88	68.50	3.549	2.72
HiAR [37]	82.82	65.32	62.42	70.19	3.758	1.60
Causal Forcing [36]	89.26	59.58	62.61	70.48	3.589	4.36
Self Forcing [9]	85.35	61.49	**66.33**	71.06	3.719	2.21
Trajectory Forcing (Ours)	**90.39**	**65.46**	66.08	**73.98**	**3.771**	**1.38**
Rolling Forcing [13]	82.04	65.36	66.77	71.39	3.804	0.56
LongLive [29]	81.33	66.07	67.04	71.48	3.801	0.60

The improvement spans both Quality *and* Semantic relative to the reported baseline results. Self Forcing reports Quality (85.07) and Semantic (81.28), while Causal Forcing reports Semantic (81.84) and Quality (84.59). Trajectory Forcing improves both, consistent with our analysis in §3.2, where the denoising trajectory provides richer cross-chunk information than a single endpoint and enables gains in both Quality and Semantic without sacrificing either. Intuitively, the selection function adapts the conditioning fidelity to the current denoising step: early steps receive coarser trajectory points that preserve spatial layout without imposing premature detail, while later steps receive more refined points that guide texture synthesis. Endpoint-only conditioning provides the same fully detailed reference at every step, which can over-constrain early steps and suppress dynamic content. The ablation in §4.4 supports our interpretation that this gain stems from trajectory structure rather than generic perturbation.

4.3 Long-Video Results

A 30-second video extends the autoregressive chain to six times the training horizon, amplifying the effect of conditioning design on error accumulation. Among directly comparable methods (CF, SF, and Ours), Trajectory Forcing achieves the highest Total (73.98), with the strongest Temporal Quality score (90.39 vs. CF 89.26 and SF 85.35) while remaining competitive on Frame Quality (65.46). This is consistent with the motivation in §3.2: over a long chain, trajectory context transfers richer information than a single endpoint and better resists the accumulation of visual errors. Quality Drift further quantifies error accumulation as the absolute imaging-quality gap between the first and last segments. Trajectory Forcing achieves Quality Drift of 1.38, about 62% of that of SF (2.21) and a third of CF (4.36). The VideoScore2 text-alignment sub-scores are 3.771 for TraF, 3.719 for SF, and 3.589 for CF. Text Alignment scores are comparable between TraF and SF (Ours 66.08, SF 66.33), indicating that the primary benefit of trajectory context lies in temporal and visual fidelity rather than text alignment. This is consistent with the trajectory-level formulation in Eq. (2): each additional chunk in the autoregressive chain contributes a full trajectory of intermediate predictions, and discarding these in favor of a single endpoint forfeits an increasingly large portion of the generation history. In the 30-second setting, which extends the chain to six times the training horizon, Trajectory Forcing achieves a higher Total score and lower Quality Drift than SF and CF.

At a broader level, the results suggest that Temporal Quality trades off against Frame Quality and Text Alignment at long horizons. Causal Forcing achieves high Temporal Quality (89.26) but the lowest Frame Quality (59.58), while LongLive and Rolling Forcing show the opposite pattern, achieving higher Frame Quality but substantially lower Temporal Quality, consistent with near-static generation that avoids temporal errors at the cost of dynamic richness. These two methods benefit from long-video-specific training, which is orthogonal to the conditioning design studied in this work. Trajectory Forcing achieves the highest Total without such additions and mitigates this trade-off, as trajectory conditioning supplies enough cross-chunk information to keep Temporal Quality high while preserving Frame Quality (65.46, comparable to RF 65.36 and LongLive 66.07).

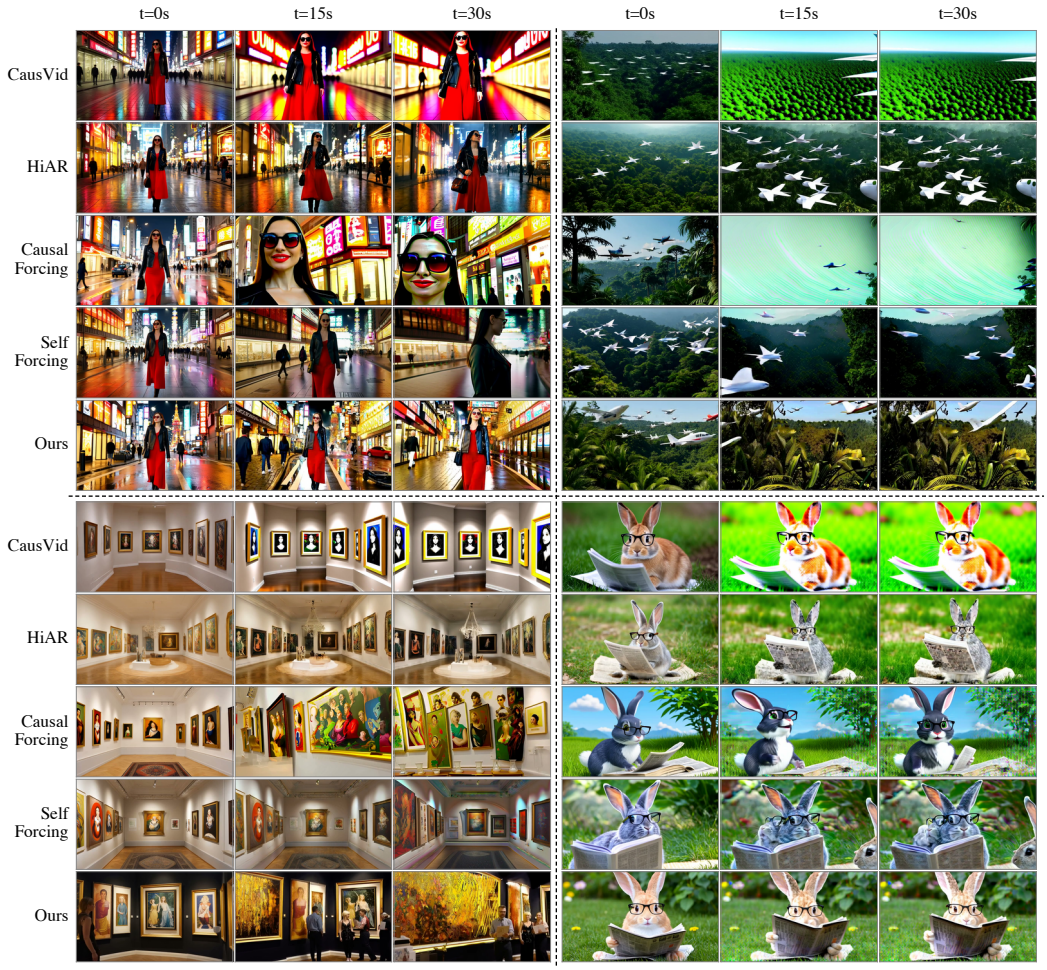


Figure 3: **Qualitative comparison on 30-second videos.** Three frames are sampled at t=0 s, t=15 s, and t=30 s for four representative MovieGen prompts: stylish woman (top-left), paper airplanes (top-right), art gallery tour (bottom-left), and rabbit reading newspaper (bottom-right). Trajectory Forcing (Ours) maintains subject coherence and color stability across the full 30 seconds, whereas Causal Forcing exhibits noticeable color drift after t=15 s, HiAR produces visually stable but near-static outputs, Self Forcing shows gradual detail loss in later segments, and CausVid degrades earliest, with pronounced color drift and texture artifacts by t=15 s.

4.4 Ablation Studies

We conduct three ablations to analyze the selection function, the source of improvement, and the contribution of each training stage.

Selection function. Figure 2 shows an ablation of the selection parameter γ across the full pipeline. For this ablation, we train and evaluate a separate model for each tested γ. Every tested finite γ outperforms the endpoint baseline in Total (γ=∞, Total 84.24), with γ=2–3 forming a broad optimum (Total 85.15 and 85.13). Step-aligned selection (γ=1, Total 84.82) underperforms this optimum, suggesting that a trajectory point at the same fidelity as the current step provides limited additional information. The optimum at moderate γ is consistent with the coarse-to-fine principle in §3.2, where the selected point should lead the current step in signal quality by a moderate margin.

Noise vs. trajectory. An alternative explanation is that any degradation of the conditioning signal may suffice to prevent dynamic collapse. To test this, we replace trajectory predictions with endpoint predictions corrupted by isotropic Gaussian noise at two magnitudes: σ=0.22 (calibrated to match

Table 4: **Ablation studies** (5-second VBench). *Left:* noise vs. trajectory comparison, where endpoint predictions with added Gaussian noise of varying magnitude replace trajectory predictions. *Right:* SL-stage comparison across three conditioning strategies under identical conditions (architecture, data, 20K steps), evaluated with 50-step denoising before distillation.

Conditioning	Total↑	Quality↑	Semantic↑
Gaussian (σ=0.22)	83.87	84.76	80.29
Gaussian (σ=0.11)	83.95	84.92	80.08
Trajectory	**85.15**	**85.89**	**82.20**

SL stage	Total↑	Quality↑	Semantic↑
Teacher Forcing	83.47	84.09	81.02
Diffusion Forcing	84.07	84.66	**81.68**
Trajectory	**84.39**	**85.25**	80.96

the empirical deviation between trajectory predictions and ground truth) and σ=0.11 (a moderate level). Table 4 (left) shows that both Gaussian variants score below the endpoint baseline reported in Figure 2 (Total 84.24), while Trajectory reaches 85.15. Gaussian corruption is not merely weaker than Trajectory—it is worse than using no perturbation at all. These Gaussian controls do not reproduce the gain, supporting our interpretation that trajectory predictions provide useful conditioning information beyond the endpoint.

SL stage. The first stage of our pipeline (§3.1) trains with pseudo-trajectory conditioning before distillation. Table 4 (right) compares three conditioning strategies under identical conditions, evaluated before distillation. Trajectory achieves the highest Total (84.39) and Quality (85.25), outperforming Teacher Forcing by +0.92 and Diffusion Forcing by +0.32 in Total. Trajectory produces lower Semantic (80.96) than Diffusion Forcing (81.68), likely because frequent conditioning on degraded predictions reduces the model's reliance on fine-grained spatial information. This effect is specific to the pre-distillation stage: in the full pipeline (Table 2), distillation recovers semantic precision, and the final model achieves the highest Semantic (82.20) among all compared methods.

4.5 Qualitative Comparison

Figure 3 shows keyframes from 30-second videos on representative MovieGen prompts. Trajectory Forcing maintains subject coherence and color stability across the full 30 seconds, while other methods exhibit varying degrees of color drift, detail loss, or temporal stagnation. Motion richness and temporal coherence are best assessed via video.

5 Discussion and Conclusion

Research on autoregressive video diffusion has focused on distillation objectives [32, 31], training paradigms for exposure bias [9, 36, 1], and inference architectures [12, 37]. This work starts from a different observation: in autoregressive diffusion, each chunk is generated through a multi-step denoising process that produces a trajectory of increasingly refined predictions. The trajectory constitutes the full generation signal of each chunk, yet existing methods reduce it to a single endpoint. Existing conditioning strategies—teacher forcing, diffusion forcing, and self-forcing—can each be understood as conditioning on ground-truth, forward-corrupted ground-truth, and predicted endpoints, respectively (Table 1). We show that conditioning on this trajectory—rather than discarding it—improves both short- and long-video quality while remaining compatible with existing distillation and inference frameworks.

Limitations. The current work studies a single base model (Wan2.1-1.3B) at a single resolution and frame rate. Verifying generalization across architectures, resolutions, and a broader range of denoising step counts remains an important direction for future work. Combining trajectory conditioning with long-video training strategies (§4.3) is a natural extension that we have not yet explored. Under the reported setting, TraF improves quality at lower throughput than Self Forcing (15.0 vs. 17.0 FPS); we have not evaluated an exactly matched-throughput operating point.

Conclusion. Trajectory Forcing implements trajectory-level conditioning in both training stages: pseudo-trajectories constructed from ground-truth videos in the SL stage, and real trajectories produced by autoregressive rollout in the RL stage. The method achieves the highest scores on both short-video (85.15 Total) and long-video (73.98 Total) benchmarks among directly comparable

methods. On 30-second generation, which extends six times beyond the training horizon, Trajectory Forcing achieves the lowest Quality Drift among directly comparable methods (Table 3). These results suggest that the denoising trajectory is a rich and previously untapped signal for autoregressive video generation.

Acknowledgments and Disclosure of Funding

This work received no external funding. The authors declare no competing financial interests beyond their listed primary affiliations.

References

[1] 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.

[2] 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.

[3] 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.

[4] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. *arXiv preprint arXiv:2412.14169*, 2024.

[5] Google DeepMind. Veo 3 technical report, 2025. Technical report.

[6] Yuwei Guo, Ceyuan Yang, Hao He, Yang Zhao, Meng Wei, Zhenheng Yang, Weilin Huang, and Dahua Lin. End-to-end training for autoregressive video diffusion via self-resampling. *arXiv preprint arXiv:2512.15702*, 2025.

[7] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[8] Xuan He, Dongfu Jiang, Ping Nie, Minghao Liu, Zhengxuan Jiang, Mingyi Su, Wentao Ma, Junru Lin, Chun Ye, Yi Lu, Keming Wu, Benjamin Schneider, Quy Duc Do, Zhuofeng Li, Yiming Jia, Yuxuan Zhang, Guo Cheng, Haozhe Wang, Wangchunshu Zhou, Qunshu Lin, Yuanxing Zhang, Ge Zhang, Wenhao Huang, and Wenhu Chen. Videoscore2: Think before you score in generative video evaluation, 2025. URL https://arxiv.org/abs/2509.22799.

[9] 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.

[10] 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.

[11] 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. *arXiv preprint arXiv:2410.05954*, 2024.

[12] Jinxiu Liu, Xuanming Liu, Kangfu Mei, Yandong Wen, Ming-Hsuan Yang, and Weiyang Liu. Streaming autoregressive video generation via diagonal distillation, 2026. URL https://arxiv.org/abs/2603.09488.

[13] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025.

[14] 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.

[15] OpenAI. Video generation models as world simulators, 2024. Technical report.

[16] Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models, 2025. Google DeepMind Blog.

[17] Ryan Po, Eric Ryan Chan, Changan Chen, and Gordon Wetzstein. Bagger: Backwards aggregation for mitigating drift in autoregressive video diffusion models. *arXiv preprint arXiv:2512.12080*, 2025.

[18] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[19] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[20] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. *arXiv preprint arXiv:2502.06764*, 2025.

[21] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[22] 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.

[23] Ruisi Wang, Zhongang Cai, Fanyi Pu, Junxiang Xu, Wanqi Yin, Maijunxian Wang, Ran Ji, Chenyang Gu, Bo Li, Ziqi Huang, Hokin Deng, Dahua Lin, Ziwei Liu, and Lei Yang. Demystifying video reasoning. *arXiv preprint arXiv:2603.16870*, 2026.

[24] Wenhao Wang and Yi Yang. VidProM: A million-scale real prompt-gallery dataset for text-to-video diffusion models. In *Advances in Neural Information Processing Systems*, volume 37, pages 65618–65642, 2024. doi: 10.52202/079017-2096.

[25] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. MIT Press, Cambridge, MA, 1949.

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

[27] Xiaofei Wu, Guozhen Zhang, Zhiyong Xu, Yuan Zhou, Qinglin Lu, and Xuming He. Pack and force your memory: Long-form and consistent video generation. *arXiv preprint arXiv:2510.01784*, 2025.

[28] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

[29] 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.

[30] Deheng Ye, Fangyun Zhou, Jiacheng Lv, Jianqi Ma, Jun Zhang, Junyan Lv, Junyou Li, Minwen Deng, Mingyu Yang, Qiang Fu, et al. Yan: Foundational interactive video generation. *arXiv preprint arXiv:2508.08601*, 2025.

[31] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. In *NeurIPS*, 2024.

[32] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

[33] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22963–22974, 2025.

[34] 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.

[35] Zeyu Zhang, Shuning Chang, Yuanyu He, Yizeng Han, Jiasheng Tang, Fan Wang, and Bohan Zhuang. Blockvid: Block diffusion for high-quality and consistent minute-long video generation. *arXiv preprint arXiv:2511.22973*, 2025.

[36] 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.

[37] Kai Zou, Dian Zheng, Hongbo Liu, Tiankai Hang, Bin Liu, and Nenghai Yu. Hiar: Efficient autoregressive long video generation via hierarchical denoising. *arXiv preprint arXiv:2603.08703*, 2026.

[38] Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. OpenVid-1M: A large-scale high-quality dataset for text-to-video generation. *arXiv preprint arXiv:2407.02371*, 2024.

A Wiener-Inspired Selection Rule

We derive the selection function S_γ using an SNR-guided heuristic motivated by the Wiener filtering perspective outlined in §3.2.

Setup. Under the flow matching forward process $x_t = (1-t)\,x_0 + t\,\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$, the minimum mean-square error (MMSE) estimate of x_0 given x_t is $\hat{x}_0 = \mathbb{E}[x_0 \mid x_t]$. Under a linear-Gaussian signal model, this estimate acts as a Wiener filter [25] whose effective bandwidth is determined by the signal-to-noise ratio.

Following the standard convention, the signal-to-noise ratio at noise level t is the ratio of signal variance to noise variance:

$$\mathrm{SNR}(t) \triangleq \frac{\alpha_t^2}{\sigma_t^2} = \frac{(1-t)^2}{t^2}. \tag{6}$$

From variance to amplitude. While SNR is canonically defined in power (variance) units, we parameterize the lookahead in amplitude (standard deviation) units. We use the v-prediction parameterization, $v_t = \dot{\alpha}_t\,x_0 + \dot{\sigma}_t\,\epsilon$. The ratio α_t/σ_t compares the signal and noise scaling coefficients in the forward interpolation. We adopt this amplitude SNR as a heuristic measure of how much one denoising step "leads" another:

$$\sqrt{\mathrm{SNR}(t)} = \frac{\alpha_t}{\sigma_t} = \frac{1-t}{t}. \tag{7}$$

In terms of denoising progress $p = 1 - \sigma \in [0, 1]$, this becomes $\sqrt{\mathrm{SNR}(p)} = p/(1-p)$.

Derivation. At denoising step j with progress p_j, the current generation operates at amplitude SNR $p_j/(1-p_j)$. We seek a context trajectory point whose amplitude SNR leads the current step by a factor $\gamma \geq 1$:

$$\sqrt{\mathrm{SNR}(p_c)} = \gamma \cdot \sqrt{\mathrm{SNR}(p_j)}. \tag{8}$$

Substituting and cross-multiplying:

$$\frac{p_c}{1-p_c} = \gamma \cdot \frac{p_j}{1-p_j} \qquad \Longrightarrow \qquad p_c = \frac{\gamma\,p_j}{1+(\gamma-1)\,p_j}. \tag{9}$$

This yields the selection function $S_\gamma(p) = \gamma p \,/\, [1 + (\gamma - 1)p]$ stated in Eq. 4.

Properties. **No lookahead ($\gamma = 1$).** $S_1(p) = p$, so the context step p_c is exactly equal to the current step p_j. There is no "lead" or "lookahead" between context and query, and the scheme reduces to step-aligned conditioning.

Degenerate lookahead ($\gamma \to \infty$). $S_\gamma(p) \to 1$ for all $p > 0$, so the context step p_c is pushed all the way to 1, corresponding to the fully clean signal x_0, as in Self Forcing [9] and Causal Forcing [36].

Balanced lookahead ($\gamma = 2$). This setting provides a moderate lead: for $0 < p < 1$, the continuous target progress $S_2(p)$ has nominal amplitude SNR exactly double that of the current progress p, without forcing endpoint-only conditioning. The iterative refinement structure is preserved. Consistent with this analysis, $\gamma \in [2, 3]$ forms a broad empirical optimum (Figure 2).

Discrete trajectory selection. For finite-grid selection, the regularized target amplitude SNR and progress are

$$r_c = \gamma \frac{p}{1-p} + (\gamma - 1)\delta_T, \qquad p_c = \frac{r_c}{1+r_c}.$$

We select the available trajectory point whose progress is nearest to p_c. For uniform pre-shift noise levels $1 - j/T$, $j = 0, \ldots, T-1$, with shift factor 5, we set $\delta_T = 2/[3(10T-9)]$. For a fixed denoising grid and γ, the selected indices can be precomputed, while the retrieved trajectory predictions remain sample-dependent.

B Training Details

Our training pipeline consists of two stages, both built on the open-source Wan2.1-T2V-1.3B [21] codebase with causal attention. For the SL stage, we use a filtered subset of OpenVid-1M [38]. The RL settings and model initializations are described in §4.1. Table 5 summarizes the configuration.

Table 5: Training configuration. The settings above the first divider refer to the RL stage.

Component	**Setting**
Base model	Wan2.1-T2V-1.3B (causal attention)
Video resolution	832×480, 16 fps, 81 frames (5 s)
RL prompts	VidProM [24] ($\sim$250K after filtering)
RL prompt rewriting	Qwen2.5-7B-Instruct [19]
Optimizer	Adam (lr=2×10^{-6}, β_1=0, β_2=0.999)
Batch size	64
Noise schedule	Flow matching with time-step shifting (k=5)
Inference steps	4 (following Self Forcing [9])
Stage 1: Ground-truth supervision (§3.1)	
Training steps	20K
Supervision	Flow matching velocity loss
Conditioning	Pseudo-trajectory (§3.1)
Stage 2: Teacher supervision (§3.2)	
Teacher (real score)	Wan2.1-T2V-14B
Critic (fake score)	Initialized from 1.3B base model
Training steps	2K
Supervision	DMD [32] (reverse KL)
Conditioning	Trajectory (§3.2)
Gradient truncation	Single random step per rollout

Noise schedule. Following Wan2.1, we adopt the flow matching framework with time-step shifting $t'(k,t) = kt\,/\,(1+(k-1)t)$ and shift factor $k=5$. The forward process is $x_t = (1-t)\,x_0 + t\,\epsilon$ with $\epsilon \sim \mathcal{N}(0,I)$ and $t \in [0,1]$. The model predicts velocity $v_\theta(x_t,t)$, and the clean-signal estimate is $\hat{x}_0 = x_t - t\cdot v_\theta(x_t,t)$.

Prompt preprocessing. In the RL stage, we use the VidProS subset of VidProM [24], filtering prompts shorter than 20 characters, containing command-line arguments, or with NSFW probability above 0.01. This yields approximately 250K prompts, which are rewritten with Qwen2.5-7B-Instruct [19] using the system prompt from the Wan2.1 release. VBench evaluation prompts are similarly rewritten for consistency.

Auxiliary predictor diagnostic. On 32 clips (192 chunk-samples, chunks 2–7), we fix the prompt, preceding ground-truth context, and noise ϵ within each chunk, and separately query the Stage-1 predictor at $\sigma \in \{0.25, 0.50, 0.625, 0.833, 0.9375\}$. The mean pairwise RMS distances between these estimates, normalized by the RMS of x_0, range from 3.7% to 19.7%. This diagnostic is separate from the fixed-velocity analytical construction in Algorithm 1.

DMD details. In Stage 2, we write the DMD gradient schematically, omitting the weighting and noise-schedule factors: $\nabla_\phi \mathcal{L}_{\mathrm{DMD}} = -\mathbb{E}_{t,\hat{x}_t}[(s_{\mathrm{real}}(\hat{x}_t,t) - s_{\mathrm{fake}}(\hat{x}_t,t))\,\partial G_\phi/\partial\phi]$, where s_{real} is the frozen Wan2.1-14B teacher and s_{fake} is a learned critic initialized from the 1.3B base model. We use the DMD objective adopted by Self Forcing [9]. The inference window covers the full five-second training context. Under the default four-step Stage-2 recipe, trajectory-cache maintenance adds an estimated 6% to training time relative to the matched endpoint-only loop, with no additional denoiser evaluations.

C Evaluation Details

5-second VBench. We evaluate all 16 VBench [10] dimensions using 946 prompts. Quality score aggregates 7 dimensions (subject consistency, background consistency, temporal flickering, motion smoothness, dynamic degree, aesthetic quality, imaging quality), where dynamic degree receives weight 0.5 and all others receive weight 1.0. Semantic score aggregates 9 dimensions (object class, multiple objects, human action, color, spatial relationship, scene, temporal style, overall consistency, appearance style), all equally weighted. Total is computed as $(4\times\text{Quality}+\text{Semantic})\,/\,5$ on a 0–100 scale. Models evaluated by us use rewritten prompts when supported.

Results for LTX-Video, Wan2.1-1.3B, MAGI-1, NOVA, Pyramid Flow, SkyReels-V2, CausVid, and Causal Forcing are taken from Causal Forcing [36]. Self Forcing results are taken from its original paper [9]. HiAR, Rolling Forcing, and LongLive results are obtained from our evaluation.

Repeatability. In each run, the endpoint-only baseline ($\gamma = \infty$) and TraF start from the same SL checkpoint. Their VBench Total scores are 84.24/85.15 in the first run and 84.33/85.13 in the second run, respectively.

Ten-step transfer. We retrain matched trajectory-conditioned and endpoint-only DMD models at $T = 10$ under two training seeds. Within each pair, both arms share the initialization, optimization settings, uniform ten-step schedule, and frozen Wan2.1-1.3B real-score teacher; both use a 1,000-iteration budget and the final checkpoint. We retain $\gamma = 2$ without retuning. On the full 946-prompt VBench protocol, endpoint-only DMD/TraF Total scores are 83.26/84.33 and 83.44/84.45, giving a mean gain of +1.04 (paired prompt-bootstrap 95% CI: [+0.17, +1.90]). These experiments use a 1.3B teacher rather than the 14B teacher in the four-step setting, so we compare matched within-setting gains rather than absolute scores across step counts.

Non-distilled ten-step inference. In a separate evaluation, we hold the non-distilled SL checkpoint fixed and vary only γ under a standard ten-step Flow Matching ODE solver. On the full 946-prompt VBench protocol, Total scores are 83.51, 83.90, 83.77, and 83.20 for $\gamma = 1, 2, 3, \infty$, respectively. The default $\gamma = 2$ improves over endpoint conditioning by +0.70 (paired prompt-bootstrap 95% CI: [+0.17, +1.17]).

30-second evaluation. We generate 30-second videos using the first 128 prompts, in file order, from the public 1,003-prompt MovieGenBench [18] list distributed with Self Forcing [9]. Each 30-second video consists of 126 latent frames, split into 6 segments of 5 seconds each. We evaluate 9 VBench dimensions in `custom_input` mode on each segment and average across segments. The remaining 7 semantic dimensions (object class, multiple objects, human action, color, spatial relationship, scene, appearance style) are excluded because they require dimension-specific prompt templates not available in the MovieGen prompt set. Following CausVid [33], the 9 evaluated dimensions are grouped as **Temporal Quality** (5 dims: subject consistency, background consistency, temporal flickering, motion smoothness, dynamic degree), **Frame Quality** (2 dims: aesthetic quality, imaging quality), and **Text Alignment** (2 dims: temporal style, overall consistency), with $\text{Total} = (\text{Temporal Quality} + \text{Frame Quality} + \text{Text Alignment})\,/\,3$. All raw scores are normalized to a 0–100 scale using the same normalization constants as Causal Forcing [36]. Treating prompts as the independent units, we perform 10,000 paired bootstrap resamples over the 128 prompts after aggregating each prompt's six segments. The 30-second Total gain over Self Forcing is +2.92 (95% CI: [+2.06, +3.78]), quantifying uncertainty from prompt sampling.

Table 6: Score differences and 95% paired prompt-bootstrap confidence intervals for 30-second evaluation.

Metric	TraF − Self Forcing	TraF − Causal Forcing
Total	$+2.92\,[+2.06, +3.78]$	$+3.50\,[+2.76, +4.26]$
Temporal Quality	$+5.04\,[+3.62, +6.47]$	$+1.13\,[+0.23, +2.01]$
Frame Quality	$+3.97\,[+3.07, +4.89]$	$+5.88\,[+5.01, +6.75]$
Text Alignment	$-0.25\,[-2.03, +1.59]$	$+3.47\,[+1.77, +5.26]$

Inference configuration for 30-second videos. All block-first methods (SF, CF, and Ours) use identical sliding-window inference with `local_attn_size=21` and `sink_size=3`, where the first chunk is permanently retained in the KV cache as global context [28, 13]. EMA weights are used for all methods. HiAR uses its native step-first denoising with pipeline parallelism. Rolling Forcing uses its rolling diffusion window. LongLive uses its short-window attention with frame sink.

VideoScore2. We use VideoScore2 [8] with the Qwen2.5-VL backbone to evaluate text alignment. Each 5-second segment is scored independently, and the text-alignment sub-score is averaged across all 6 segments.

D Broader Impacts

Trajectory Forcing may support creative production, animation, and educational visualization through the synthesis of extended video sequences. These capabilities also create risks: synthetic videos could be used to fabricate events, misrepresent real people, or spread misleading visual narratives. Longer sequences may make fabricated content more persuasive when detached from its original context. Responsible use calls for clear identification of synthetic content and careful consideration of consent, privacy, and intellectual property. We encourage downstream users to assess these risks in their intended applications and avoid presenting generated examples as documentary evidence.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the trajectory-level conditioning formulation, two-stage training pipeline, and experimental results, all supported by §4.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in §5, including single base model and unexplored combination with long-video training.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The selection formula is motivated by an SNR-guided heuristic. The algebra in Appendix A does not establish an optimality result for video diffusion.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide full training and evaluation details in §4.1, Appendix B, and Appendix C.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: An anonymized implementation was provided for peer review. The code will be publicly released through the project page once internal approval is obtained.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the experiment details in §4.1 and Appendix B.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We evaluate on two benchmarks (5s VBench, 30s MovieGenBench) with multiple metrics and conduct ablations across γ values, conditioning sources, and SL-stage strategies. Paired prompt-bootstrap confidence intervals are reported in Appendix C.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Company policy restricts disclosure of detailed compute-resource usage. We report the training and evaluation settings used for fair comparisons.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors confirm that this research complies with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential positive and negative societal impacts in Appendix D.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No model weights or datasets are released with this version. Research code will be released through the project page after internal review and approval.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers for all models, datasets, and tools used. The official model releases of Wan2.1-T2V-1.3B, Wan2.1-T2V-14B, and Qwen2.5-7B-Instruct, and the VBench evaluation code, specify Apache-2.0. The OpenVid-1M dataset card lists CC BY 4.0, describes the dataset as intended for research and non-commercial use, and directs users to follow the source-video licenses. The VidProM prompts are listed under CC BY-NC 4.0.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new research code, model weights, or datasets are publicly released with this version. Research code will be released through the project page after internal review and approval.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing or experiments with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve experiments with human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use Qwen2.5-7B-Instruct for prompt rewriting, as described in §4.1.