THE SERIALITY GAP IN VIDEO DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

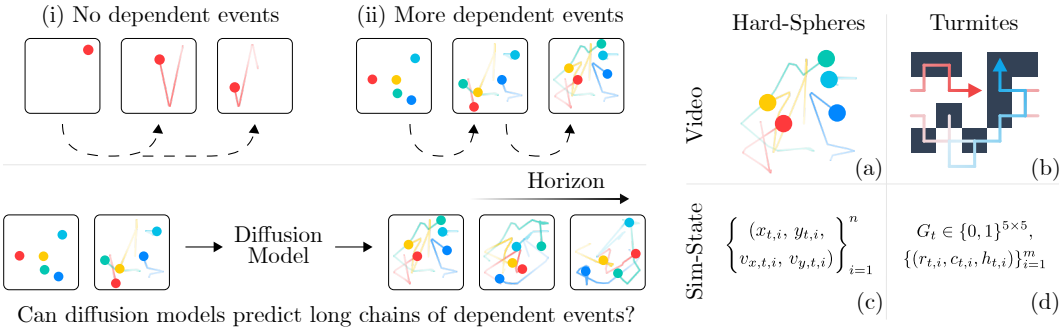


Figure 1: **Dependent-event prediction exposes the seriality gap.** (Top left) Hard-sphere dynamics separate non-serial from serial video prediction. (i) In the single-ball control, any future state can be computed directly from the initial state, without resolving intermediate states. (ii) With multiple balls, each ball–ball collision changes the state governing later collisions, creating dependent-event chains that must be resolved in temporal order. (Bottom left) Given initial frames, can a diffusion model remain accurate as longer prediction horizons demand more serial computation? (Right) We evaluate hard-sphere dynamics and Turmites under video and simulator-state prediction.

ABSTRACT

When one ball strikes another, then another, video models should predict the consequences of each bounce. In controlled experiments on multi-ball hard-sphere dynamics, we find that the performance of standard bidirectional video diffusion degrades as the causal chain lengthens, even when provided more denoising steps. In a length-matched control, where ball–ball interactions are absent, the degradation largely disappears, isolating dependent-event structure rather than video length as the cause. Across intervention studies, **methods that increase effective serial computation improve performance disproportionately**, including autoregressive/blockwise generation and architectural depth. This finding extends to both video prediction and simulator-state prediction, and also applies to both physical and algorithmic tasks. We identify this pattern as the **seriality gap**: a mismatch between tasks requiring growing serial computation and video diffusion models whose denoising loop does not provide scalable serial compute. We then prove that, for fully observed deterministic video prediction with an exact score, denoising steps do not add serial computation beyond the backbone, indicating a structural obstacle for video diffusion on serial reasoning and simulation tasks.

1 INTRODUCTION

When one ball strikes a second, which then strikes a third, predicting the chain of consequences seems like a minimal competence for any video model.[1] Yet, even as video diffusion models generate compelling clips and are increasingly positioned as world simulators and visual reasoners (Brooks et al., 2024; Wiedemer et al., 2025), persistent failures in physical consistency (Meng et al., 2024; Guo et al., 2025; Motamed et al., 2025; Rädsch et al., 2026; Tragoudaras et al., 2026; Huang et al., 2024; Zheng et al., 2025) and reasoning (Wiedemer et al., 2025; Newman et al., 2026) remain. What

[1] Interactive project page: `https://seriality-gap-anon-site-v2.vercel.app`

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.

if this is due not only to a lack of data coverage but also to a limitation of the diffusion model itself? We therefore ask the following:

Can video diffusion models predict ever-longer chains of dependent events?

This simple question is consequential precisely because the *negative* answer would be surprising. Because diffusion generates videos through repeated denoising, failure on this task would call into question what computation those iterations actually provide—an issue that has recently received substantial attention but remains unresolved. Some studies attribute reasoning to computation across denoising steps (Wang et al., 2026b); others report erosion of motion structure over later steps (Han et al., 2026), early plan commitment and the need for chained generations for longer solutions (Newman et al., 2026), or rapidly saturating gains from additional steps (Ma et al., 2025). Together, these findings motivate characterizing whether denoising supplies computation that scales with dependency-chain length (Section 6).

We begin with a deliberately minimal video-prediction task: **hard-sphere dynamics**. Multiple identical balls move frictionlessly in a box and undergo elastic collisions; given the initial frames of a trajectory, the model must predict the future video (Figure 1). Here, visual complexity is minimal, the dynamics are deterministic, and prediction errors are unambiguous. Longer rollouts contain increasingly deep chains of *non-trivial* ball–ball interactions. With only ball–wall interactions, as in the single-ball and separate-lane experiments, the trajectories admit a closed-form solution: prediction does not need to go through the intermediate events. Our setup thereby lets us vary serial complexity while holding visual content essentially fixed, a control difficult to obtain in natural video (Section 2).

The task's simplicity is deliberate. Natural video adds appearance complexity, uncertainty, and ambiguity, but none removes the need to propagate dependent events. Failure in this controlled setting therefore identifies a bottleneck that may persist in richer settings; conversely, success would address that bottleneck, not solve video modeling as a whole.

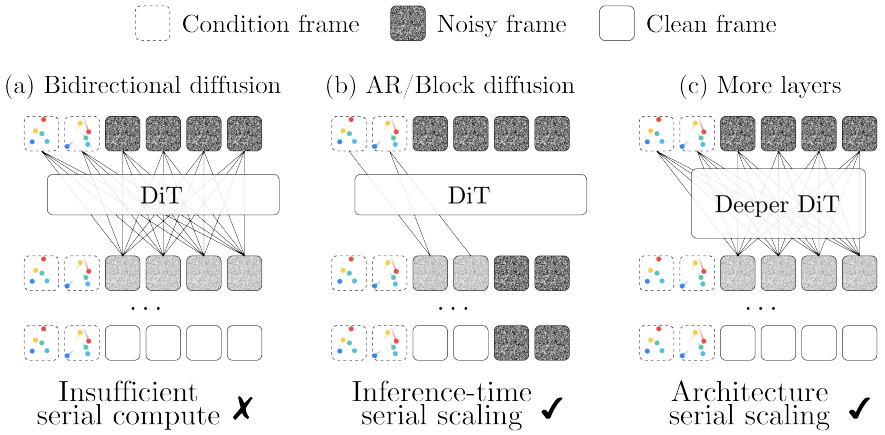


Figure 2: **Serial scaling.** (a) Bidirectional diffusion jointly denoises future frames, but denoising does not supply scalable serial compute (Section 4). Consistent with our theory, performance improves with (b) autoregressive/blockwise inference factorization or (c) a deeper backbone.

The seriality gap (Section 3) In controlled multi-ball experiments, standard *bidirectional* video diffusion degrades relative to autoregression as measured dependency depth increases, even at fixed horizons. This degradation largely disappears for a single ball and in separate-lane multi-ball controls, both without ball–ball interactions. This suggests that the non-trivial ball–ball event dependencies explain the degradation, rather than video length or object count alone. Increasing denoising steps provides little benefit and does not close the performance gap. By contrast, at comparable total compute, interventions that add effective serial computation help disproportionately: increasing depth is more effective than increasing width, and serializing the generation—from bidirectional generation through progressively smaller temporal blocks to latent-frame autoregression—yields almost monotonic improvements. Figure 2 depicts the bidirectional diffusion, inference serialization, and increased depth models.

Why denoising steps do not close the performance gap (Section 4) Despite their step-by-step form, additional denoising iterations do not close the performance gap. There are arguments on both sides about whether denoising steps should help. On one hand, prior work finds that performance gains from additional diffusion steps plateau in image generation (Ma et al., 2025). On the other hand, Wang et al. (2026b) argue that reasoning in video models happens primarily along the denoising steps. Furthermore, the community has proposed noise rescheduling (Qiu et al., 2024) and temporal augmentation (Chen et al., 2025) to improve video coherence. In Proposition 4.1, we show that, for fully observed deterministic prediction with an exact score, the seemingly step-by-step denoising loop is a single-step prediction in disguise: it adds no scalable computational depth beyond the backbone.

Beyond hard-sphere video The seriality gap is not specific to hard-sphere dynamics or video prediction. It can arise whenever diffusion models face tasks whose serial demands grow with problem size, including maze and puzzle solving. Our experiments cover both video and simulator-state prediction on physical tasks such as hard-sphere dynamics and algorithmic tasks such as **Turmites**, where an agent reads a cell's color, turns, flips the color, and moves; each action can affect later actions. The seriality gap manifests similarly in all cases: more diffusion steps do not close the gap, while more sequential inference helps. This is, however, not to say that latent-frame autoregression is a universal remedy. Not all tasks have dependencies that follow video-frame order. Maze solving is one example where most computation may be paid up front. The needed serial computation must instead come from greater backbone depth, a task-aligned inference factorization, or potentially a new class of non-score-matching generative models.

Contributions

(i) We introduce controlled testbeds for dependent-event prediction in hard-sphere dynamics and Turmites, varying serial complexity while keeping visual content simple (Section 2; Appendix C.2).

(ii) We identify the *seriality gap* across tasks and representations: the performance of bidirectional diffusion degrades as chains of dependent events lengthen, despite increasing total compute. This degradation is not explained by video length or closed by additional denoising steps, but is reduced by interventions that add effective serial computation (Section 3).

(iii) We show theoretically that, for fully observed deterministic video prediction with an exact score, denoising steps do not add scalable serial computation beyond the backbone (Section 4).

2 Hard-Sphere Dynamics as One Testbed for Dependent-Event Prediction

In order to study serial temporal prediction, we design a setting where dependency length can vary while visual complexity and ambiguity remain controlled. We use **hard-sphere dynamics** (Figure 1 (left, top)): n identical balls move frictionlessly in a square box and undergo perfectly elastic collisions with walls and with one another. The rendering is simple, the dynamics are deterministic, and each initial condition has a single ground-truth rollout. This deliberately minimal setting is diagnostic—if a video model fails here, the failure is unlikely to be explained by visual complexity or ambiguous futures.

In this setting, serial complexity is determined by dependency depth: the longest chain of dependent ball–ball collisions, which can be shorter than the total. This lets us compare multi-ball ($n > 1$) rollouts, where collision chains grow with the prediction horizon, against single-ball ($n = 1$) rollouts, which can be equally long but have closed-form, non-serial futures. This contrast separates difficulty due to video length from difficulty due to serial complexity. The experiments also extend to direct simulator-state prediction and Turmites, a discrete algorithmic task (Appendix C).

2.1 Experiment Settings

Dataset For each hard-sphere video dataset configuration, we synthesize f-frame hard-sphere simulation videos at 128×128 resolution. Because the simulator is deterministic, each initial condition has a unique ground-truth rollout, so prediction error can be measured directly and unambiguously. Appendix B documents the simulator, initial-condition sampling, and filtering criteria.

We consider interacting ball trajectories and non-interacting controls. For the interacting five-ball training sets, a post-conditioning ball–ball collision-density filter sets a horizon-dependent minimum event count. The matched evaluation clips are not filtered at each horizon; their observed mean collision count and dependency depth nevertheless grow with f (Appendix C.3.1). Since each ball–ball collision updates the state that determines subsequent collisions, these events must be resolved in temporal order. We track dependency depth directly from each trajectory's collision-dependency graph (Appendix C.3.1); average depth grows with horizon, supporting f as a proxy for serial complexity. In the length-matched control ($n = 1$), ball–ball collisions are absent, so any future state can be computed in closed form from the initial conditions without simulating intermediate states. In addition, we confine five balls to separate lanes, removing ball–ball interactions while preserving object count and horizon.

As f increases, bidirectional video diffusion models process larger spatiotemporal tensors and therefore receive more total computation (Wan et al., 2025). If this computation were sufficient for tracking dependent events, local physical accuracy in the multi-ball setting should not degrade substantially more with f than in the non-interacting controls.

Models We train DiT-style (Peebles & Xie, 2023) video diffusion models adapted from `Wan2.1` (Wan et al., 2025), using its VAE-compressed video latents. With the same base architecture, we compare bidirectional denoising, Block-k generation with a block width of k temporal VAE latents, and autoregressive generation one latent at a time (Figure 2). Each generated block uses the same flow-matching sampler, with $T = 50$ denoising steps by default. We also vary backbone depth and width; direct simulator-state experiments bypass rendering and VAE compression. For hard-sphere video, reported TFLOPs and sequential latency are analytic estimates; TFLOPs exclude the VAE and assume cached prefixes. Appendix B specifies the latent representation, sampler, and compute accounting, while Appendix H gives the attention masks.

Evaluation We evaluate predictions along three axes.

1. **Global error** $\Delta x^{(\mathrm{GT})}$ measures the average Euclidean distance between predicted and ground-truth ball centers across all balls and evaluated frames. This is the most direct notion of prediction accuracy, but it is sensitive to accumulated long-horizon drift.
2. **Local error** $\Delta x^{(k)}$ measures local physical consistency. For each generated frame, we go back k frames, extract ball centers and velocities from the generated pixels, roll the simulator forward for k steps, and compare the result to the generated frame. This avoids penalizing accumulated drift caused by state-estimation errors. By default, we report $\Delta x^{(5)}$.
3. **Visual quality (IoU)** measures shape fidelity independently of position error. For each ball, we center-align its generated silhouette with an ideal circular mask of the correct radius and compute their intersection over union.

We score only the $f - f_0$ future frames, with $f_0 = 5$; the local metric may rewind into the conditioning window. Appendix B defines the metrics and state-extraction procedure.

3 The Seriality Gap in Hard-Sphere Dynamics and Turmites

We summarize the empirical findings as five observations. Appendix I gives full results; Appendix F rules out optimization and initialization confounds.

Observation 1 Bidirectional diffusion degrades as videos get longer. In the $n = 5$ training sets, the collision-density filter sets a minimum event count that grows with f. The matched evaluation sets are not filtered at each horizon, but their mean measured dependency depth grows with f (Appendix C.3.1). Measured dependency depth increases with f, and the performance gap widens with measured depth (Appendix C.3). Although both bidirectional and autoregressive models receive more inference compute at longer horizons, with temporal self-attention scaling roughly as $\mathcal{O}(f^2)$, autoregressive generation maintains lower and nearly flat local physical error $\Delta x^{(5)}$. Bidirectional diffusion instead worsens with f, as shown in Figure 3 (top left). The paired bidirectional–autoregressive gap intervals exclude zero at both model widths and all four horizons (Table 4).

Observation 2 The gap disappears without ball–ball collisions. As a length-matched control, we set $n = 1$, removing ball–ball collisions and leaving trajectories that can be computed in closed form from the initial state. In this non-serial setting, bidirectional and autoregressive models have nearly identical local physical error $\Delta x^{(5)}$, and both remain stable as f increases, as shown in Figure 3 (top left). The gap also vanishes when five balls are confined to separate lanes, preserving object count and horizon while removing ball–ball interactions (Appendix I). Together, these controls show that video length alone cannot explain the degradation and support serial complexity as its key driver.

Observation 3 Serializing inference improves performance. We interpolate from bidirectional to autoregressive diffusion using block-causal masks over temporal latent frames, keeping the backbone fixed. Larger blocks denoise more frames jointly, while smaller blocks impose a more sequential generation order. Local physical error improves as the factorization becomes more serial, following bidirectional (Block-11) $\rightarrow$ Block-3 $\rightarrow$ Block-2 $\approx$ autoregressive (Block-1) across model sizes, as shown in Figure 3 (bottom right). Since model capacity is unchanged, these gains reflect temporal compute allocation rather than a larger backbone. The improvement from Block-3 through Block-2 to autoregression also holds with one shared set of hard-sphere model weights (Appendix F.1). The serialization advantage persists across alternative state extractors and extraction-free image metrics (Appendix B.4.2).

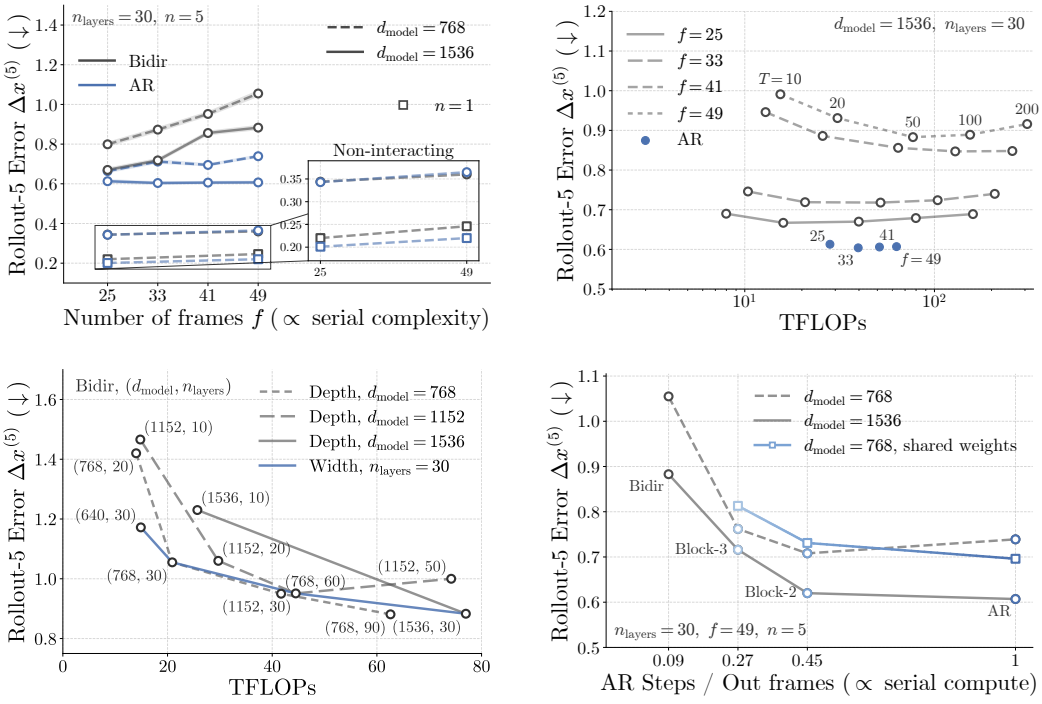


Figure 3: **Empirical evidence for the seriality gap.** (Top left) Observations 1 and 2. (Top right) Observation 4. (Bottom left) Observation 5. (Bottom right) Observation 3. All plotted errors are $\Delta x^{(5)}$ (lower is better). Shaded regions indicate uncertainty.

Observation 4 More denoising steps do not close the gap. A natural explanation is that bidirectional diffusion may simply be under-computed at inference time. If denoising iterations supplied the serial computation needed to predict longer collision chains, increasing T should especially improve long-horizon rollouts. We therefore evaluate bidirectional diffusion with $T \in \{10, 20, 50, 100, 200\}$ across all video lengths. Performance improves up to roughly $T = 50$; beyond this point, additional denoising steps provide no consistent benefit and sometimes increase error, while the long-horizon degradation remains. Increasing T therefore does not close the gap to autoregressive generation, as shown in Figure 3 (top right).

Observation 5 Depth scaling is more effective than width scaling. We next ask whether the missing serial computation can be supplied by the backbone itself. We sweep bidirectional models

over width d_{model} and depth n_{layers}, varying how capacity and compute are allocated. Bidirectional models benefit more strongly from increasing depth than increasing width up to about 30 layers. At comparable budgets, deeper, narrower models outperform shallower, wider ones, and the gain from $n_{\text{layers}} = 10$ to 30 layers exceeds the improvements from widening a 30-layer model from $d_{\text{model}} = 640$ to 1536, as shown in Figure 3 (bottom left). Beyond 30 layers, additional depth gives diminishing returns, possibly because deeper models become harder to optimize (Srivastava et al., 2015).

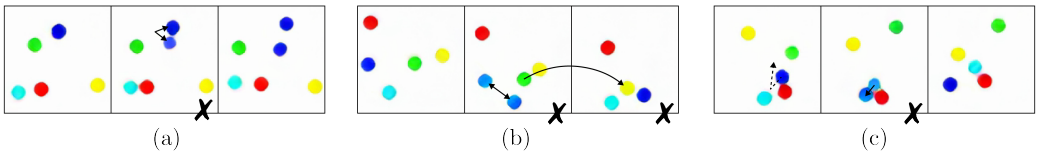


Figure 4: **Illustrative failure modes in bidirectional generations.** We show common qualitative failures produced by the bidirectional diffusion model trained on $f = 49$, $n = 5$, $d_{\text{model}} = 1536$, and $n_{\text{layers}} = 30$. (a) The dark-blue ball splits into two, resulting in a ball-count violation. (b) The dark-blue and light-blue balls exchange colors, while the green and yellow balls merge into one; both failures violate state consistency. (c) The dark-blue ball should bounce off the light-blue ball, but instead tunnels through it. Additional qualitative examples, including side-by-side comparisons with autoregressive generations, are shown in Appendix D.

Qualitative failure modes We observe three recurring state-consistency failures, illustrated in Figure 4: tunneling, ball-count violations, and color-identity violations. In tunneling, a ball passes through another object or disappears and reappears elsewhere without a physically valid intermediate trajectory. In ball-count violations, two balls merge into one or one ball splits into multiple balls. In color-identity violations, a ball changes color, duplicates another ball's color, or swaps its color identity with another ball. These failures can occur in both bidirectional and autoregressive generations, but they are much more frequent in bidirectional models, especially for high-velocity balls and long multi-ball rollouts.

In the backbone depth-width ablation, all model sizes produce visually plausible videos, but smaller and especially shallower backbones exhibit these state-consistency failures more often.

3.1 Across Tasks and Representations

Direct simulator-state prediction The trends on hard-sphere dynamics generalize beyond video. When predicting coordinates and velocities directly, the performance gap widens with horizon, and more denoising steps do not close it. At $f = 32$ frames, bidirectional accuracy peaks at $T = 1$, consistent with the account developed in Section 4: with no pixels to synthesize, additional denoising steps provide little to no benefit, and no scalable serial computation for resolving longer dependency chains. Thus, Observations 1 and 4 hold without rendering, VAE compression, or state extraction (Appendix C.1).

Turmites The same trends hold on a discrete algorithmic visual-reasoning task—distinct from continuous hard-sphere simulation. In both video and direct simulator-state prediction, the performance gap widens with the number of predicted turns, and additional denoising steps do not close it. In simulator state, autoregressive generation maintains 1.000 final-grid accuracy while bidirectional accuracy falls from 1.000 at 9 turns to 0.496 at 39 turns, and more serial blockwise factorizations approach autoregressive accuracy. With one checkpoint trained on all four factorizations, accuracy follows AR $>$ Block-5 $>$ Block-15 $>$ bidirectional at every tested horizon (Appendix F.1). Thus, Turmites reproduce Observations 1 and 4 in both representations and Observation 3 in simulator state (Appendix C.2).

4 Why Denoising Steps Do Not Provide Scalable Serial Computation

The observations above are counterintuitive. As videos get longer, bidirectional diffusion receives roughly $\mathcal{O}(f^2)$ compute from attention, even though the simulation only requires $\mathcal{O}(f)$ compute. So

the accuracy declines despite the model having plenty of compute. This points to a failure to convert such compute into usable serial computation for predicting dependent events.

Through a complexity-theoretic lens, the findings are not so puzzling. Video diffusion models are asked to solve an inherently serial problem, but their denoising loops do not add scalable serial computation. We first introduce the following complexity classes.

1. At the *parallelizable* end, the class TC^0 captures polynomial-size, constant-depth threshold circuits, closer to a closed-form shortcut than to sequential simulation.
2. At the *serial* end, P-complete problems capture polynomial-time computations believed not to be efficiently parallelizable.[2]

On one hand, we task the model with inherently serial problems. Hard-sphere collisions form dependent chains similar to the classical billiard-ball computation which is P-complete (Fredkin & Toffoli, 1982; Greenlaw et al., 1995). Turmite cell flips can affect later turns, and a general Turmite prediction problem is also P-complete (Maldonado et al., 2017). These results motivate our tasks but do not establish hardness for our benchmark instances.

On the other hand, our fixed-depth transformer backbones lie in TC^0 under log-precision assumptions (Merrill & Sabharwal, 2023). Under the assumption above, a TC^0 backbone cannot solve a general P-complete prediction problem by itself. Can repeated denoising supply the missing serial computation? *No.* Following, we show why it cannot in the exact-score setting of fully observed deterministic video prediction.

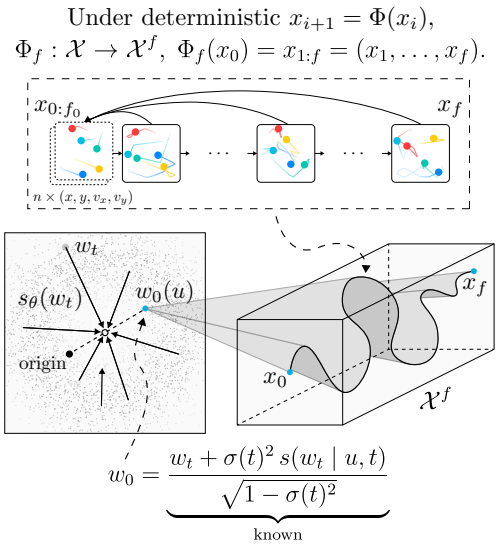


Figure 5: **Theory overview.** Deterministic dynamics map an initial state s_0 to a unique future trajectory through repeated transitions $s_{i+1} = \Phi(s_i)$. Given s_0, the conditional distribution over futures is therefore a point mass. Under Gaussian noising, the exact conditional score at any fixed noise level strictly between zero noise and pure noise points back to this unique future, so *one* score-network evaluation can recover it (eq. (3)). Because the clean future is recovered from a single score-network evaluation, denoising steps do not add scalable serial computation beyond what the backbone already computes.

Under deterministic dynamics, the initial state determines a unique future. If the conditioning frames determine that state, a single exact-score evaluation can recover the future (Figure 5). This connects the denoising argument to the complexity classes above:

1. Under the assumptions below, if a TC^0 network computes the exact score, then state prediction can be solved in TC^0 (Proposition 4.1).
2. State prediction can be P-hard, as general billiard-ball computation illustrates (Fredkin & Toffoli, 1982; Greenlaw et al., 1995).
3. Under $\mathsf{P} \neq \mathsf{NC}$, such a P-hard family lies outside TC^0 and cannot have that exact-score predictor.

Therefore, more denoising steps cannot overcome the general case of state prediction problem.

4.1 Exact-Score Recovery for Deterministic Video Prediction

We now formalize how one score evaluation recovers the future. For a deterministic process, let F map a physical state x_t to its video frame $y_t = F(x_t)$, and let the fixed conditioning window $y_{0:f_0}$

[2]Here, "efficiently parallelizable" means solvable in polylogarithmic parallel time with polynomial total resources. Without this restriction, serial depth can often be reduced by an exponential blow-up in width, which is not considered an efficient parallel algorithm (Greenlaw et al., 1995). The separation from P-complete problems relies on standard conjectures such as $\mathsf{P} \neq \mathsf{NC}$, which are widely believed but unproved.

suffice to recover the state. We call this assumption *full observability*. We assume finite-precision encodings and that F and finite-window state recovery are logspace-uniform TC^0. This lets us render the initial states as conditioning frames and recover the final state from the predicted frames within TC^0. At horizon f, the *video prediction task* is to predict the future frames $y_{f_0+1:f}$ from the conditioning frames $y_{0:f_0}$. The *state prediction task* is to predict the final state x_f from the initial states $x_{0:f_0}$ (Appendix A.1).

We use the noise-conditional score network (NCSN) formalism, which connects to other diffusion formulations (Luo, 2022) and is detailed in Appendix A. For a fixed prediction horizon f, let $u := y_{0:f_0}$ denote the conditioning video and $w_0 := y_{f_0+1:f}$ its future. At a fixed noise level $0 < \sigma(t) < 1$, Gaussian noising takes the form

$$w_t = \sqrt{1-\sigma(t)^2}\, w_0 + \sigma(t) z, \qquad z \sim \mathcal{N}(0, I). \tag{1}$$

Let $\rho(\cdot \mid u, t)$ denote the conditional density of w_t given u. Define the score function

$$s(w_t \mid u, t) := \nabla_{w_t} \log \rho(w_t \mid u, t). \tag{2}$$

An NCSN uses a network s_θ to approximate this score. The exact score is generally unavailable in closed form. When $w_0 = w_0(u)$ is deterministic, however, the conditional score admits a closed-form expression. A single exact-score evaluation then recovers $w_0(u)$ as

$$w_0(u) = \frac{\sigma(t)^2 s(w_t \mid u, t) + w_t}{\sqrt{1-\sigma(t)^2}}. \tag{3}$$

For the finite-precision result, one exact-score evaluation at a fixed noise level with rational coefficients suffices; recovering $w_0(u)$ then involves only logspace-uniform TC^0 arithmetic (Appendix A.3).

For complexity arguments, we use the associated output-bit decision problem. We ask for one bit of the predicted state at a time so that the task has a yes/no answer. Given $(x_{0:f_0}, 1^f, j)$, determine the j-th bit of the encoded final state x_f. All P-hardness claims are under logspace many-one reductions.

Proposition 4.1 *Under the finite-precision and state-recovery assumptions above, if an NCSN in logspace-uniform TC^0 computes the exact conditional score for the video prediction task, then the state prediction task is in logspace-uniform TC^0.*

Proof. We solve the state prediction problem by an algorithm that evaluates the NCSN just once. The entire algorithm runs in TC^0. The proof is in Appendix A.3. □

One discrepancy from practice is that our trained models approximate the score. Scaling model size and training compute should improve approximation, bringing the learned models closer to the exact-score setting of Proposition 4.1. Yet even with an exact score, denoising adds no scalable serial computation beyond a TC^0 backbone. Our result motivates exploring objectives beyond score matching that encourage each generation step to build on earlier computation.

5 Discussion and Limitations

Implications of the seriality gap At comparable computation, scaling serial computation consistently show greater gains for predicting dependent events on both videos and states and in both physical and algorithm tasks. Our theoretical result identifies a structural reason: under fully observed deterministic dynamics, one exact-score evaluation suffices to recover the future. The backbone must therefore already be capable of solving the prediction problem in a single evaluation since additional denoising cannot supply the missing computational capacity.

This finding may help explain early plan commitment in maze solving (Newman et al., 2026) and performance plateaus with additional denoising steps in image generation (Ma et al., 2025). Limited serial computation may also contribute to the generalization failures reported by Kang et al. (2025), although generalization might still be a problem even with sufficient computation. In particular, Wang et al. (2026b) interpret evolving predictions as reasoning across denoising steps. Our findings question whether denoising steps can supplies the serial computation needed for reasoning at all.

Autoregression is not a universal fix Autoregression helps in our tasks because generating successive states follows their event dependencies. Other visual reasoning tasks need not factor this way. Maze solving, for example, can require exploring alternative routes before even choosing the first move; the amount of search need not track the number of video frames. The generation order therefore needs to match the task dependencies; more frames or autoregression alone need not supply the required computation.

More serial computation helps only if the model learns to use it to track event dependencies. In natural video, models must also represent appearance and uncertainty. These demands may favor visually plausible outputs over accurate dynamics. Our controlled testbeds reduce these confounds. Whether the same interventions improve dependent-event prediction in natural video remains open.

Limitations Our theoretical result assumes fully observed deterministic dynamics and an exact score. Whether this result extends to stochastic or partially observed dynamics remains open. In our preliminary vibrating-wall experiment (Appendix E), stochastic wall collisions make the future uncertain, yet autoregression remains more locally consistent than bidir. Our theoretical result also does not explain every source of the performance gap: training limitations may also cause this gap even when sufficient serial computation is available.

6 Related Work

Video models as simulators and reasoners Video generators are increasingly studied as world simulators and visual reasoners (Brooks et al., 2024; Wan et al., 2025; Wiedemer et al., 2025; Wang et al., 2026a; Bai et al., 2025; Gillman et al., 2026). Recent benchmarks evaluate physical consistency and commonsense, including comparisons with real-world experiments and checks of physical laws (Zheng et al., 2025; Rädsch et al., 2026; Tragoudaras et al., 2026). Synthetic physics environments provide a complementary way to remove visual realism as a confound: Kang et al. (2025) study video diffusion models in the PHYRE physical simulator (Bakhtin et al., 2019), finding strong in-distribution performance but failures to generalize physical laws to out-of-distribution settings. We use hard-sphere dynamics and Turmites to isolate the effect of dependency depth on prediction accuracy, using simple visuals and direct simulator-state prediction to control for visual complexity.

Video diffusion inference paradigms Bidirectional video diffusion denoises all frames of a clip jointly. Autoregressive and sliding-window methods generate successive frames or blocks, enabling streaming and longer videos (Weng et al., 2024; Xie et al., 2025; Yin et al., 2025). We treat these inference choices not only as engineering tradeoffs, but as different ways of allocating serial computation across time.

Serial computation and depth Our analysis builds on limits of transformer parallelism and depth–width tradeoffs (Merrill & Sabharwal, 2023; Telgarsky, 2016; Chen et al., 2024). The Serial Scaling Hypothesis (Liu et al., 2026) derives a TC^0 bound for diffusion from sampling convergence under a bounded intrinsic dimension assumption. Applying this bound to video prediction requires justifying that the dimension remains bounded as the prediction horizon grows. We instead exploit fully observed deterministic dynamics: the conditioning frames determine a unique future, however long the video. This gives a concrete physical setting in which we can establish the computational limit directly, without a separate dimension assumption or a sampling-convergence argument.

Other temporal bottlenecks in video diffusion Prior work improves temporal quality through noise rescheduling (Qiu et al., 2024), training objectives that jointly predict appearance and motion (Chefer et al., 2025), temporal regularization (Chen et al., 2025), and preservation of motion from early denoising steps (Han et al., 2026). PhyCo combines physical-property conditioning with VLM-guided reward optimization to improve physical consistency and control (Narayanan et al., 2026). These methods can improve motion and coherence, but improvements that leave effective serial computation unchanged do not address the bottleneck we identify. Our experiments show that deeper backbones and more sequential generation help resolve longer dependencies; additional denoising steps do not.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with implementing research code, including code used to create figures. We have not used generative AI tools for developing theory or proofs, designing experiments, analyzing data, or interpreting results. Additionally, we used generative AI tools for drafting, editing, and polishing parts of the manuscript; finding relevant literature; and preparing citations. We have reviewed all AI-assisted work. The authors used these tools interactively, manually reviewed and iteratively revised the AI-assisted text, revised most AI-assisted code, inspected the figures, and checked the cited sources. AI was not used as an autonomous research agent. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our human evaluation was a small, voluntary, uncompensated preference check on synthetic videos. The interface did not request identifying or demographic information. We present the selected-example results only as a limited perceptual check; further study details, including review status, appear in Appendix G.

REPRODUCIBILITY STATEMENT

The appendices provide the simulator and dataset construction, model training and evaluation procedures, and the assumptions and proof of our theoretical result. We will provide a link to the code and release the trained model checkpoints after the review process.

REFERENCES

Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, May 1982. ISSN 0304-4149. doi: 10.1016/0304-4149(82)90051-5. URL http://dx.doi.org/10.1016/0304-4149(82)90051-5.

Yutong Bai, Danny Tran, Amir Bar, Yann LeCun, Trevor Darrell, and Jitendra Malik. Whole-body conditioned egocentric video prediction. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 164375–164418. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f0552f14388d95b19740dee809f5cad1-Paper-Conference.pdf.

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/4191ef5f6c1576762869ac49281130c9-Paper.pdf.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Hila Chefer, Uriel Singer, Amit Zohar, Yuval Kirstain, Adam Polyak, Yaniv Taigman, Lior Wolf, and Shelly Sheynin. VideoJAM: Joint appearance-motion representations for enhanced motion generation in video models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7595–7616. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chefer25a.html.

Harold Haodong Chen, Haojian Huang, Xianfeng Wu, Yexin Liu, Yajing Bai, Wen-Jie Shu, Harry Yang, and Ser-Nam Lim. Temporal regularization makes your video generator stronger, 2025. URL https://arxiv.org/abs/2503.15417.

Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer transformer, 2024. URL https://arxiv.org/abs/2412.02975.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2(3):4, 2024.

Nicolas Dufour, Alexei A. Efros, and Patrick Pérez. The fid lottery: Quantifying hidden randomness in generative-model evaluation, 2026. URL https://arxiv.org/abs/2606.20536.

Edward Fredkin and Tommaso Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3-4):219–253, April 1982. ISSN 1572-9575. doi: 10.1007/bf01857727. URL http://dx.doi.org/10.1007/BF01857727.

Nate Gillman, Yinghua Zhou, Zitian Tang, Evan Luo, Arjan Chakravarthy, Daksh Aggarwal, Michael Freeman, Charles Herrmann, and Chen Sun. Goal force: Teaching video models to accomplish physics-conditioned goals. *arXiv preprint arXiv:2601.05848*, 2026. URL https://arxiv.org/abs/2601.05848.

Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, June 1995. ISBN 9780197560518. doi: 10.1093/oso/9780195085914.001.0001. URL http://dx.doi.org/10.1093/oso/9780195085914.001.0001.

Xuyang Guo, Jiayan Huo, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Jiale Zhao. T2vphysbench: A first-principles benchmark for physical consistency in text-to-video generation, 2025. URL https://arxiv.org/abs/2505.00337.

Woojung Han, Seil Kang, Youngjun Jun, Min-Hung Chen, Fu-En Yang, and Seong Jae Hwang. Physics in 2-steps: Locking motion priors before visual refinement erases them. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.06361.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL http://dx.doi.org/10.1038/s41586-020-2649-2.

William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002. doi: 10.1016/S0022-0000(02)00025-9. URL https://www.cs.rutgers.edu/~allender/papers/division.pdf.

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 (CVPR)*, pp. 21807–21818, June 2024.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 28991–29017. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/kang25g.html.

Hao Li, Yang Zou, Ying Wang, Orchid Majumder, Yusheng Xie, R. Manmatha, Ashwin Swaminathan, Zhuowen Tu, Stefano Ermon, and Stefano Soatto. On the scalability of diffusion-based text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9400–9409, June 2024.

Yuxi Liu, Konpat Preechakul, Kananart Kuwarananchaoen, and Yutong Bai. The serial scaling hypothesis, 2026. URL https://arxiv.org/abs/2507.12549.

Calvin Luo. Understanding diffusion models: A unified perspective, 2022. URL https://arxiv.org/abs/2208.11970.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025. URL https://arxiv.org/abs/2501.09732.

Diego Maldonado, Anahí Gajardo, Benjamin Hellouin de Menibus, and Andrés Moreira. Nontrivial turmites are turing-universal. *arXiv preprint arXiv:1702.05547*, 2017.

Fanqing Meng, Jiaqi Liao, Xinyu Tan, Wenqi Shao, Quanfeng Lu, Kaipeng Zhang, Yu Cheng, Dianqi Li, Yu Qiao, and Ping Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation, 2024. URL https://arxiv.org/abs/2410.05363.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. ISSN 2307-387X. doi: 10.1162/tacl_a_00562. URL http://dx.doi.org/10.1162/tacl_a_00562.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles?, 2025. URL https://arxiv.org/abs/2501.09038.

Sriram Narayanan, Ziyu Jiang, Srinivasa Narasimhan, and Manmohan Chandraker. PhyCo: Learning controllable physical priors for generative motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41892–41902, 2026. URL https://arxiv.org/abs/2604.28169.

Kaleb Newman, Tyler Zhu, and Olga Russakovsky. Video models reason early: Exploiting plan commitment for maze solving, 2026. URL https://arxiv.org/abs/2603.30043.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023.

Haonan Qiu, Menghan Xia, Yong Zhang, Yingqing He, Xintao Wang, Ying Shan, and Ziwei Liu. Freenoise: Tuning-free longer video diffusion via noise rescheduling, 2024. URL https://arxiv.org/abs/2310.15169.

Tim Rädsch, Yuki M. Asano, Hilde Kuehne, Stefan Bauer, Priyank Jaini, Robert Geirhos, and Carsten T. Lüth. Physics-iq verified, 2026. URL https://arxiv.org/abs/2606.18943.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/215a71a12769b056c3c32e7299f1c5ed-Paper.pdf.

Matus Telgarsky. benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir (eds.), *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/telgarsky16.html.

Antonios Tragoudaras, Chenyu Zhang, Daniil Cherniavskii, Antonios Vozikis, Thijmen Nijdam, Derck W. E. Prinzhorn, Mark Bodracska, Nicu Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Evaluating newtonian mechanics in video generative models with real physical systems, 2026. URL https://arxiv.org/abs/2504.02918.

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

Maijunxian Wang, Ruisi Wang, Juyi Lin, Ran Ji, Thaddäus Wiedemer, Qingying Gao, Dezhi Luo, Yaoyao Qian, Lianyu Huang, Zelong Hong, Jiahui Ge, Qianli Ma, Hang He, Yifan Zhou, Lingzi Guo, Lantao Mei, Jiachen Li, Hanwen Xing, Tianqi Zhao, Fengyuan Yu, Weihang Xiao, Yizheng Jiao, Jianheng Hou, Danyang Zhang, Pengcheng Xu, Boyang Zhong, Zehong Zhao, Gaoyun Fang, John Kitaoka, Yile Xu, Hua Xu, Kenton Blacutt, Tin Nguyen, Siyuan Song, Haoran Sun, Shaoyue Wen, Linyang He, Runming Wang, Yanzhi Wang, Mengyue Yang, Ziqiao Ma, Raphaël Millière, Freda Shi, Nuno Vasconcelos, Daniel Khashabi, Alan Yuille, Yilun Du, Ziming Liu, Bo Li, Dahua Lin, Ziwei Liu, Vikash Kumar, Yijiang Li, Lei Yang, Zhongang Cai, and Hokin Deng. A very big video reasoning suite. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2602.20159.

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. In *Proceedings of the European Conference on Computer Vision*, 2026b. URL https://arxiv.org/abs/2603.16870.

Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, Chong Luo, Yueyi Zhang, and Zhiwei Xiong. Art•v: Auto-regressive text-to-video generation with diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 7395–7405. IEEE, June 2024. doi: 10.1109/cvprw63382.2024.00735. URL http://dx.doi.org/10.1109/CVPRW63382.2024.00735.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners, 2025. URL https://arxiv.org/abs/2509.20328.

Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 6376–6386, June 2025.

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

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness, 2025. URL https://arxiv.org/abs/2503.21755.

A Theoretical Details

A.1 Problem Formalism

Suppose we are given a *deterministic* process. Let the state space be X, and let the deterministic time evolution be $\Phi : X \to X$. We have an observation function $F : X \to Y$, which takes a state $x \in X$ and converts it into an image $F(x) \in Y$. Here, Y is the space of possible images. For example, if we consider an image to be an $H \times W$ array of pixels, then $Y = \mathbb{R}^{H \times W}$.

We consider a finite-precision problem family indexed by input length of n bits. Elements of X and Y, including all state coordinates and pixel values, are represented using $K = O(\log n)$ bits per scalar. Our outputs have polynomial length. We say they are computable in TC^0 when each indexed output bit is computable in TC^0. The input includes the unary horizon 1^f ($f > f_0$); state and frame dimensions are polynomial in n. We consider valid trajectories and assume that the dynamics and rendering preserve the $O(\log n)$-bit representation.

Suppose that the physical process is *fully observable*. That is, we assume that there exists a constant number f_0 such that, given any two trajectories

$$x_0, x_1 = \Phi(x_0), x_2 = \Phi(x_1), \ldots; \quad x'_0, x'_1 = \Phi(x'_0), x'_2 = \Phi(x'_1), \ldots,$$

if $F(x_0) = F(x'_0), F(x_1) = F(x'_1), \ldots F(x_{f_0}) = F(x'_{f_0})$, then $x_0 = x'_0$.

In other words, we can fully determine x_0 if we observe $f_0 + 1$ consecutive frames $F(x_0), F(x_1), \ldots, F(x_{f_0})$.

We additionally assume that the observation map F and finite-window state recovery are logspace-uniform TC^0. With a slight abuse of notation, we write the latter as

$$F^{-1}\left(F(x_t), \ldots, F(x_{t+f_0})\right) = x_{t:t+f_0}.$$

The key point is simple. For deterministic video prediction, once the conditioning frames determine the state, there is only one correct future video. After Gaussian noising, the conditional distribution is centered on a scaled version of that unique future, so the exact score at *any* fixed noise level strictly between zero noise and pure noise contains enough information to recover it in one step. Repeated denoising therefore does not add scalable serial computation beyond what the backbone already provides, as summarized in Figure 5.

Given this setup, we can define two formal problems:

1. **Video prediction task**: given an initial video segment $F(x_{0:f_0})$ of a physical trajectory and a number f, compute its future video segment $F(x_{f_0+1:f})$.
2. **State prediction task**: given an initial physical trajectory $x_{0:f_0}$ and a number f, compute x_f.

Suppose further that we have a diffusion model that solves the following problem: given an initial video segment $y_{0:f_0} := F(x_{0:f_0})$ of a physical trajectory and a number f, compute its future video trajectory $y_{f_0+1:f} := F(x_{f_0+1:f})$.

A.2 Diffusion Modeling

As before, let w be an object we want to sample, and let u be a conditioning input. w is an element of a sample space, here taken to be $\mathbb{R}^d$, where d is some positive integer.

The forward diffusion process is defined by a locally integrable noise rate function $\beta : [0, \infty) \to (0, \infty)$, with the condition that $\int_0^\infty \beta(t)dt = \infty$. We begin by sampling a starting point $w_0 \sim \rho_0(w|u)$ and then define the forward stochastic process:

$$w_{t+dt} = w_t - \underbrace{\frac{1}{2}\beta(t)w_t dt}_{\text{drift}} + \underbrace{\sqrt{\beta(t)}dW_t}_{\text{noise}},$$

for all $t \in [0, \infty)$, where W_t is a standard Brownian motion in $\mathbb{R}^d$. The W stands for "Wiener", since this is also called the Wiener process. It can be schematically thought of as $dW_t \sim \mathcal{N}(0, dtI_d)$.

At each fixed time, the conditional marginal distribution is:

$$w_t = \sqrt{1 - \sigma(t)^2} w_0 + \sigma(t) z, \quad z \sim \mathcal{N}(0, I_d),$$

where $\sigma(t)^2 = 1 - e^{-\int_0^t \beta(\tau) d\tau}$ is the variance due to accumulated noise.

In other words, the probability distribution of w_t, conditional on u, is the convolution of a scaled version of $\rho_0(\cdot|u)$ with a Gaussian function. The probability distributions $\rho(\cdot|u, t)$ are increasingly blurry versions of the scaled original $\rho_0(\cdot|u)$, starting at $\rho(\cdot|u, 0) = \rho_0(\cdot|u)$ and ending with $\lim_{t\to\infty} \rho(\cdot|u, t) = \rho_{\mathcal{N}(0,I_d)}$, where we lose all traces of the original and obtain the standard Gaussian distribution.

Define the score function

$$s(w_t|u, t) := \nabla_{w_t} \log \rho(w_t|u, t).$$

With the score function, the same process can be expressed in the time-reversed form (Anderson, 1982):

$$w_{t-dt} = w_t + \frac{1}{2}\beta(t) w_t dt + \beta(t) \underbrace{s(w_t|u, t)}_{\text{score function}} dt + \sqrt{\beta(t)} d\overline{W}_t.$$

Here $d\overline{W}_t$ is a Brownian increment in the reverse-time process, with covariance $dt\, I_d$ for a positive backward step dt.

In the NCSN formalism, we have a neural network $s_\theta \approx s$. We then solve the backward process by numerical stochastic integration:

$$\hat{w}_{t-\Delta t} = \hat{w}_t + \frac{1}{2}\beta(t)\hat{w}_t \Delta t + \beta(t) s_\theta(\hat{w}_t|u, t)\Delta t + \sqrt{\beta(t)}\Delta\overline{W}_t.$$

Each reverse-time increment has distribution $\Delta\overline{W}_t \sim \mathcal{N}(0, \Delta t\, I_d)$.

The exact score function s has no closed-form solution in general, because $\rho_0(w_0|u)$ is too generic to be integrated. However, if w_0 is a *deterministic* function of u, then for $0 < \sigma(t) < 1$

$$\log \rho(w_t|u, t) = -\frac{1}{2\sigma(t)^2} \left\| w_t - \sqrt{1 - \sigma(t)^2} w_0(u) \right\|^2 + \text{Const}$$

$$s(w_t|u, t) = \frac{\sqrt{1 - \sigma(t)^2} w_0(u) - w_t}{\sigma(t)^2}.$$

Thus, we are actually able to use the score function to solve for $w_0(u)$ in one step:

$$w_0(u) = \frac{\sigma(t)^2 s(w_t|u, t) + w_t}{\sqrt{1 - \sigma(t)^2}}.$$

For video prediction at horizon f, let $u := y_{0:f_0}$ and $w_0 := y_{f_0+1:f}$, with w_t its noised version. The NCSN takes (u, f, t, w_t) and approximates the score

$$s(w_t \mid u, f, t) = \nabla_{w_t} \log \rho(w_t \mid u, f, t).$$

A.3 Finite-Precision Recovery

We define the score using the continuous Gaussian density. The clean data and the inputs and outputs of the score evaluation have finite-precision encodings. We encode the noise level by its rational coefficients, rather than by the possibly irrational time $t_\star$. The NCSN is assumed to return the exact rational score at the all-zero query used below, for every valid conditioning input and horizon. Rational arithmetic uses exact binary numerator–denominator encodings with $O(\log n)$ bits, allowing a constant-factor increase for intermediate results.

For the finite-precision result, fix any noise level $t_\star$ whose coefficients $\sigma(t_\star)$ and $\sqrt{1-\sigma(t_\star)^2}$ are nonzero rationals, and query the score at $w_{t_\star} = 0$. If the NCSN computes the exact rational encoding of $s(0 \mid u, t_\star)$, then equation 3 recovers the exact finite encoding of $w_0(u)$ using only exact arithmetic on the finite encodings, which is in logspace-uniform TC^0 (Hesse et al., 2002). Thus, one score-network evaluation suffices.

Proof of Proposition 4.1. Given an instance $(x_{0:f_0}, 1^f)$ of state prediction:

1. Compute the conditioning video segment
$$y_{0:f_0} := F(x_{0:f_0}).$$
2. Fix a constant noise level $(t_\star)$ such that
$$\sigma(t_\star) = \frac{4}{5},$$
and set the noisy future input to the all-zero finite encoding,
$$\hat{y}_{f_0+1:f,t_\star} := 0.$$
3. Evaluate the exact-score NCSN once:
$$s_\star := s\left(0 \mid y_{0:f_0}, f, t_\star\right).$$
4. By equation 3,
$$y_{f_0+1:f} = \frac{\sigma(t_\star)^2 s_\star}{\sqrt{1-\sigma(t_\star)^2}} = \frac{16}{15} s_\star.$$
5. Recover the final state from the last $f_0 + 1$ available frames:
$$x_{f-f_0:f} := F^{-1}\left(y_{f-f_0:f}\right),$$
and output x_f.

By assumption, F, F^{-1}, and the exact-score NCSN are logspace-uniform TC^0. Multiplication by the fixed rational constant $16/15$ is also in logspace-uniform TC^0. Since TC^0 is closed under constant composition, state prediction is in logspace-uniform TC^0. Equivalently, each output bit is computable in logspace-uniform TC^0 on valid inputs. □

B Hard-Sphere Video Methods and Evaluation

B.1 Hard-Sphere Simulator and Datasets

Videos are synthesized with a custom NumPy Harris et al. (2020) event-driven simulator of frictionless balls in a closed square box. For ball–ball interactions, the simulator computes the next collision time analytically. For a pair of balls (i, j) with relative displacement $\Delta x = x_j - x_i$, relative velocity $\Delta v = v_j - v_i$, and radii r_i, r_j, the collision time t is the smallest nonnegative root of

$$\|\Delta x + t\Delta v\|_2^2 = (r_i + r_j)^2,$$

which yields a quadratic equation in t. Once t is found, the collision positions follow directly from $x_i(t) = x_i + v_i t$ and $x_j(t) = x_j + v_j t$.

For each rollout, we sample initial ball positions uniformly at random such that no two balls overlap at the first frame, then sample each ball's initial direction uniformly from $[0, 2\pi)$ and speed uniformly from $[8, 10]$. We then simulate a longer trajectory and extract fixed-length clips of length f frames with a sliding window, retaining windows with an overlap of at most $f/2$ to avoid highly similar neighboring clips and increase diversity. Each dataset configuration fixes the number of balls, frame length, box size, ball radius, and rendering resolution.

In the reported experiments, we use $n \in \{1, 5\}$ balls, video lengths $f \in \{25, 33, 41, 49\}$, ball radius 0.7, world size 10×10, rendering resolution 128×128, and 5 conditioning frames. Training sets contain $20{,}000$ videos. For evaluation, we generate $1{,}024$ base videos of 57 frames and derive the shorter sets by retaining only the first f frames of each video. This keeps the conditioning frames, and therefore the initial conditions, matched across frame lengths for the same sample, making cross-length comparisons fairer. For the interacting five-ball training sets, we retain only windows whose post-conditioning ball–ball collision density,

$$\frac{\text{number of ball–ball collisions}}{\text{active frames} \times \text{number of balls}},$$

exceeds 0.035. For $n = 5$, this corresponds to at least 5 collisions when $f = 33$ and at least 8 when $f = 49$. This filter is not applied to the single-ball or separate-lane controls, or to the matched evaluation clips. For the base 57-frame evaluation set, we also try to spread samples more evenly over ball–ball collision counts, although the resulting distribution is not perfectly uniform; the shorter trimmed evaluation sets inherit this coverage. Training and evaluation use disjoint random seeds and the same simulator and parameter settings. Although the evaluation prefixes have no per-horizon collision-density minimum, their mean collision count and dependency depth increase with f (Appendix C.3.1).

B.2 Video Model, Training, and Compute

Video model and latent representation Every hard-sphere video model operates in the `Wan2.1` VAE latent space, which downsamples videos by $8\times$ along each spatial axis and $4\times$ temporally. The patch embedder groups 2×2 spatial latent patches without additional temporal downsampling, so one temporal latent frame corresponds to four rendered video frames. The five conditioning frames produce two causal VAE latents.

Conditioning-state recoverability Although 96.3% of evaluation clips contain a collision somewhere in the five-frame conditioning window, 99.7% retain at least one collision-free frame interval for every ball. After a VAE encode–decode round trip, a blur-robust nearest-palette tracker recovers last-frame positions to 5.9% of ball radius. Selecting each ball's least-error velocity estimate among the four frame intervals using ground truth gives an error of 2.9% of speed; this is an oracle diagnostic, not a velocity estimator available without ground truth. Direct simulator-state prediction provides a separate test without rendering, VAE compression, or state extraction. Figure 6 illustrates the last-frame position diagnostic and the one-frame offset associated with velocity error.

Training Models are trained for $200{,}000$ iterations with a batch size of 64 using AdamW with a weight decay of 0.01 and a constant learning rate of 2×10^{-4}. We maintain an exponential moving average (EMA) of the weights with decay 0.9999 and use the EMA weights for evaluation. The reported training runs require approximately 300 A100-80GB GPU-days. This protocol describes the main hard-sphere video models, trained separately for each horizon and factorization with a common flow-matching objective and matched data within each horizon. Additional shared-weight controls jointly train causal factorizations on hard spheres and include bidirectional generation in the Turmite training mixture.

Sampler and compute accounting The implementation uses the first-order Euler update of the flow-matching scheduler. For T denoising steps, let $r_j = 1 - j/T$ and $\sigma_j = 5r_j/(1 + 4r_j)$ for $j = 0, \ldots, T$. Starting from Gaussian noise, each update is $z_{j+1} = z_j + (\sigma_{j+1} - \sigma_j)v_\theta(z_j, \sigma_j)$, with the final noise level $\sigma_T = 0$. All factorizations use this schedule; causal generation repeats it for each new block while holding the generated prefix fixed. Block-k generation absorbs a final remainder smaller than k into the preceding block. Thus, the 11 predicted latent frames at $f = 49$ form three blocks of sizes $3, 3, 5$ for Block-3. TFLOPs are analytic, batch-one estimates of DiT self-attention projections, attention products, and feed-forward layers, using two FLOPs per multiply-add. They exclude VAE encoding/decoding and embedding, normalization, and other elementwise overheads. With hidden width d, feed-forward width d_{ff}, and N tokens, one layer costs $8Nd^2 + 4N^2d + 4Ndd_{\text{ff}}$. For M new tokens attending to P cached prefix tokens, the causal estimate is $8Md^2 + 4M(P + M)d + 4Mdd_{\text{ff}}$. We sum over generated blocks and multiply by the number of layers and denoising steps. These cached-prefix estimates describe theoretical computation, not

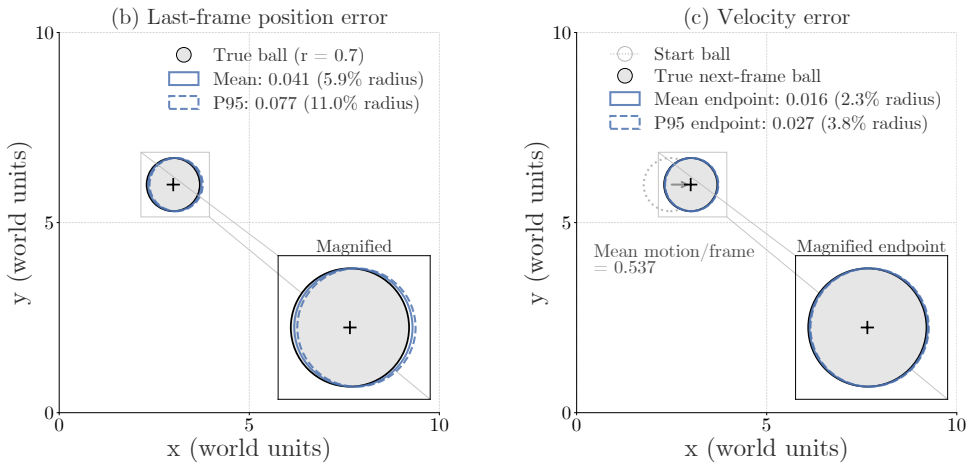


Figure 6: **Physical-state recoverability after VAE compression.** Five conditioning frames from each of 1,024 five-ball clips are encoded into two causal Wan2.1 VAE latents and decoded. Decoded centers are measured by nearest-palette tracking. Panel (b) shows last-frame center error; panel (c) converts the best of four interval-velocity errors to a one-frame endpoint offset at 15 fps. The latter uses ground truth to select the interval and is an oracle diagnostic, not a rollout. All offsets point in a schematic $+x$ direction; ball radii stay at 0.7. P95 is the per-clip 95th percentile, not a confidence interval.

measured execution time. At fixed architecture and factorization, estimates scale linearly with T. The separate sequential-latency proxy is $\tau = B\,T\,n_{\text{layers}}$, with B the actual number of generated blocks.

B.3 Evaluation Metrics

Let $\mathcal{I} = \{f_0 + 1, \ldots, f\}$ denote the predicted frames used for evaluation, with $f_0 = 5$ conditioning frames in the hard-sphere video experiments. Only the $f - f_0$ future frames are scored; the local metric may rewind into the conditioning window. We define the global and local accuracy metrics as follows:

$$\Delta x^{(\mathrm{GT})} = \frac{1}{|\mathcal{I}|\,n} \sum_{t \in \mathcal{I}} \sum_{i=1}^{n} \left\| \hat{x}_{t,i} - x_{t,i}^{\mathrm{GT}} \right\|_2, \qquad \Delta x^{(k)} = \frac{1}{|\mathcal{I}|\,n} \sum_{t \in \mathcal{I}} \sum_{i=1}^{n} \left\| \hat{x}_{t,i} - \tilde{x}_{t,i}^{(k)} \right\|_2,$$

where $\hat{x}_{t,i}$ denotes the predicted position of ball i at time t, $x_{t,i}^{\mathrm{GT}}$ the ground-truth position, and $\tilde{x}_{t,i}^{(k)}$ the position obtained by extracting the generated state at time $t - k$ and rolling the simulator forward k steps. A human evaluation in Appendix G confirms that large gaps in these metrics are salient.

B.4 State Extraction and Metric Robustness

B.4.1 State and Velocity Extraction

For evaluation, we recover ball positions using the image-moment centroid of all pixels within ± 32 in each RGB channel of its known palette color, followed by the fixed pixel-to-world transform. All matching components contribute, including disconnected pieces of a split blob; no component or identity reassignment is performed. If a mask is empty, we carry the previous extracted position forward, or use the conditioning position at the first frame. The resulting coordinates enter both $\Delta x^{(\mathrm{GT})}$ and $\Delta x^{(k)}$.

Velocity recovery uses the same event-driven solver as data generation. Using zero-based frame indices here, define $u_{t,i} = (\hat{x}_{t,i} - \hat{x}_{t-1,i})/\Delta t$, where $\Delta t = 1/\mathrm{fps}$, and initialize $\hat{v}_{0,i}$ from the conditioning velocity. At $t = 1$, the incoming velocity is $u_{1,i}$. For $t \geq 2$, it is $\hat{v}_{t-1,i}$ if ball i collided during the preceding simulated step, and $u_{t-1,i}$ otherwise. Advance one frame from the

extracted positions $\hat{x}_{t-1}$ using these incoming velocities, retaining the returned end velocities as $\hat{v}_t$ and recording which balls collided. The recovery simulation uses the solver's box-clamping option. For the local metric, start from $(\hat{x}_{t-k}, \hat{v}_{t-k})$, advance the solver for k frames, and compare its returned end positions with $\hat{x}_t$. Positions, velocities, and collision events are separate simulator outputs; the comparison uses only positions. Scored indices are $t = f_0, \ldots, f-1$ in this zero-based convention, equivalent to $\mathcal{I} = \{f_0 + 1, \ldots, f\}$ in Section 2.1. Since $f_0 = 5$ and $k \in \{1, 5\}$, every scored frame permits the full k-frame rewind. The local metric never reads future ground truth.

B.4.2 Extractor Robustness

Re-extracting ball states from artifact-free ground-truth videos gives $\Delta x^{(\mathrm{GT})} = 0.006$, $\Delta x^{(1)} = 0.014$, and $\Delta x^{(5)} = 0.095$. Table 1 separates the rollout-5 errors by extractor outcome over 225,280 scored ball-frames. Split-blob cases may overlap the mask-presence categories.

Table 1: **Rollout-5 error by extractor outcome.** Percentages and mean $\Delta x^{(5)}$ are reported separately for autoregressive and bidirectional generations.

Extractor outcome at the endpoints	AR (%)	AR $\Delta x^{(5)}$	Bidir (%)	Bidir $\Delta x^{(5)}$
Mask present at both endpoints	99.25	0.598	97.78	0.852
Absent at t only	0.36	1.505	1.07	1.533
Absent at $t-5$ only (reappears)	0.33	2.274	0.83	3.356
Absent at both	0.06	1.571	0.32	1.736
Split blob (overlapping category)	2.69	0.927	3.57	1.288
All scored ball-frames	100	0.607	100	0.883

Missing masks are more frequent in bidirectional generations, but they do not account for most of the gap: restricting to scored ball-frames with masks present at both t and $t-5$ leaves a bidirectional–autoregressive difference of 0.254, compared with 0.276 overall. Thus, about 92% of the overall difference remains in this mask-present subset. Split-blob cases can still occur in this subset, so this comparison isolates missing-mask effects rather than all visual artifacts. The distinct saturated ball colors and ± 32 per-channel tolerance prevent a color swap from arising in the extractor itself.

We test eight state extractors at each of two model widths: centroid masks with RGB tolerance from ± 8 to ± 96, median and largest-component reducers at ± 32, and tolerance-free nearest-palette assignment. Bidirectional generation has higher mean rollout-5 error than AR in all 16 settings; the 95% paired-bootstrap interval for the bidirectional-minus-AR gap excludes zero in 15 of them, with the $d_{\mathrm{model}} = 1536$, ± 8 centroid setting as the exception. The mean ordering $\{\mathrm{AR}, \text{Block-2}\} < \text{Block-3} < \mathrm{Bidir}$ holds in 14 of 16 settings. At $d_{\mathrm{model}} = 768$ with centroid tolerances ± 8 and ± 16, Block-3 instead has lower mean error than AR. Tables 5 and 6 report the means and the saved model-versus-AR intervals; those intervals do not test every adjacent step of the ordering. At $d_{\mathrm{model}} = 1536$, changing the ± 32 reducer from centroid to median changes the bidirectional-minus-AR gap from 0.2756 to 0.2489.

The comparison persists without extracting ball centers or velocities. Pixel MSE is 0.189 for AR, 0.190 for Block-2, and 0.204 for bidirectional generation (lower is better); mask IoU is 0.733, 0.732, and 0.704, respectively (higher is better). These image measures do not use the state-and-velocity extraction underlying $\Delta x^{(5)}$.

C Additional Experiments

We extend the hard-sphere video experiments to direct simulator-state prediction and to a second task, Turmites. Below, we give the experimental protocols and a fuller account of the results summarized in Section 3.

Numerical results for the comparisons in this appendix are collected in Appendix I.

C.1 Direct Hard-Sphere Simulator-State Prediction

State representation We train directly on the ground-truth simulator trajectories for $n = 5$ balls, without rendering frames or passing them through the video VAE. The conditioning input contains four frames of state, with tensor shape $4 \times 5 \times 4$ and per-ball features (x, y, v_x, v_y). The target contains the remaining future states with the same per-ball features. One temporal token represents the four-frame trajectory of one ball. This representation supplies positions and velocities exactly, removing rendering, VAE compression, and state extraction from both training and evaluation.

Model and evaluation protocol The state model is a DiT with $d_{\text{model}} = 512$, 8 layers, and 8 attention heads, trained with an x_0-prediction objective and L1 loss. Each horizon uses $200{,}000$ training trajectories. For the three-run horizon comparison, we evaluate the step-$40{,}000$ EMA checkpoints at $T = 20$ on $1{,}024$ trajectories and report means and min–max ranges across three training seeds. We compare bidirectional and autoregressive temporal factorizations over $f \in \{8, 16, 24, 32, 40, 48\}$ frames. At $f = 8$, the four conditioning frames leave one four-frame future block containing one token per ball, so the two temporal factorizations coincide. The horizon comparison reports rollout-5 drift rate in m/s (lower is better): mean rollout displacement divided by $5/15\,\text{s}$, or three times that displacement. The predicted coordinates and velocities initialize the simulator rollout. All predicted frames are scored: the first uses a four-frame rewind because only four conditioning frames are available, and subsequent frames use a five-frame rewind. The stored metric applies the same $5/15\,\text{s}$ normalization to every frame. The separate denoising-step sweep evaluates averaged step-$40{,}000$ checkpoints on 256 held-out trajectories for each factorization, horizon, and $T \in \{1, 2, 5, 10, 20, 50\}$. It reports rollout-5 drift rate in m/s, computed by dividing mean rollout displacement by $5/15\,\text{s}$.

Results Table 10 shows the horizon trend. The bidirectional–autoregressive gap grows from 0.011 at 8 frames, where the factorizations coincide, to 1.956 at 48 frames. Across the same range, the autoregressive error remains below 0.9 after $f = 8$, whereas bidirectional error rises above 2.7. At $f = 8$, separately trained models with equivalent temporal factorizations have similar errors; for $f \geq 16$, their three-run ranges do not overlap.

Increasing the denoising budget does not close the gap (Table 11). For $f \geq 16$, bidirectional performance is best at $T = 1$ or $T = 2$ and generally worsens as T increases; autoregressive performance remains substantially better at every tested budget. In this setting without pixel rendering or VAE compression, additional denoising steps do not close the measured gap, whereas temporal serialization substantially improves accuracy.

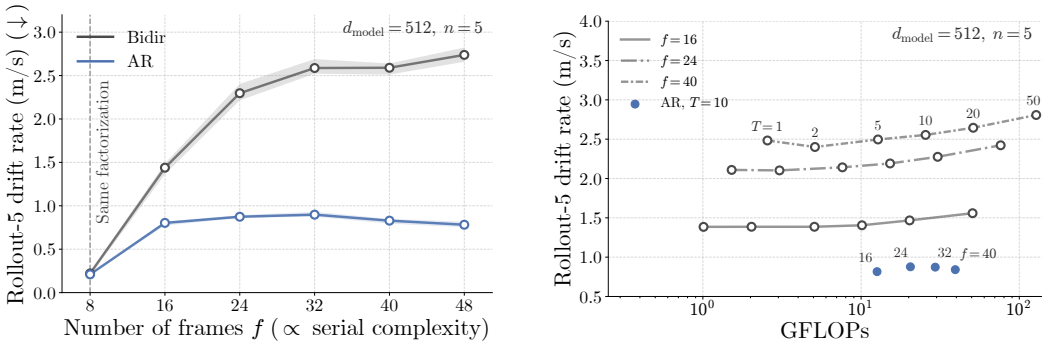


Figure 7: **The seriality gap in direct simulator-state prediction.** Left: the bidirectional–autoregressive rollout-5 drift-rate gap (m/s) grows with prediction horizon; shading spans the three runs. Right: the 256-trajectory denoising sweep reports rollout-5 drift rate (m/s) and shows that additional denoising steps do not close the gap.

C.2 TURMITES

Task and dynamics A Turmite is an agent moving on a binary grid. We use a randomly generated 5×5 light/dark grid and place the agent at a random initial location; its heading is part of the initial state. At each turn, the agent reads its current cell, turns right on a light cell or left on a dark cell, flips that cell, and moves forward one cell, wrapping around at the grid boundary. The flip changes the state that determines later turns, creating an exact chain of dependent events. General Turmite prediction is P-complete (Maldonado et al., 2017), but we use that result only as motivation: we make no hardness claim for these fixed 5×5 instances.

Video and simulator-state representations We study both rendered video and direct simulator-state prediction. The video representation renders the evolving binary grid and agent at every turn. The simulator-state representation begins with 25 grid tokens, one per cell, encoding $(\text{row}, \text{column}, \text{color})$, and one agent token encoding $(\text{row}, \text{column}, \text{heading})$; every attribute is one-hot. Tokens have 16 channels: five each for row and column, two for color, and four for heading. Active attributes are one-hot encoded, then mapped from $\{0, 1\}$ to $\{-1, +1\}$; unused channels remain at -1. Each predicted turn contains two tokens: an updated agent token $(\text{row}, \text{column}, \text{heading})$ and a cell-change token $(\text{row}, \text{column}, \text{color})$. Thus, the state representation is fully observable by construction and avoids rendering, compression, and extraction.

Model and evaluation protocol Each Turmite setting uses 20,000 training trajectories and 1,024 evaluation trajectories. The simulator-state DiT has $d_{\text{model}} = 512$, 4 layers, and 16 attention heads. It uses continuous x-prediction flow matching on the one-hot features after mapping them to $[-1, 1]$, with loss only on future tokens' active attribute channels. We compare bidirectional, autoregressive, and Block-k temporal factorizations and evaluate denoising budgets $T \in \{1, 5, 10, 20, 50\}$. The simulator-state horizon comparison uses $T = 10$. The video DiTs use $d_{\text{model}} = 768$, 10 layers, and 6 attention heads. Their 13, 25, and 49 total-frame horizons include five conditioning frames and correspond to 8, 20, and 44 predicted turns, respectively; we evaluate $T \in \{10, 20, 50, 100, 200\}$. For the video compute comparison, each temporal VAE latent has 16 spatial tokens and the feed-forward width is 3,072. We count DiT-layer FLOPs with two FLOPs per multiply-add: bidirectional generation processes all latents jointly, whereas autoregressive generation recomputes the full growing prefix for each new latent without a KV cache. The $T = 1$ per-clip counts are multiplied by T for the plotted budgets. For state prediction, after denoising each turn or block, we decode each active attribute by an independent argmax and feed the re-encoded one-hot tokens to later causal blocks; bidirectional state generation decodes after denoising the entire future. To reconstruct a state trajectory, predicted cell updates assign the decoded color to the decoded cell, and predicted agent tokens set position and heading, without enforcing the Turmite transition rule. For video prediction, we report final-board color accuracy: a cell is correct only when its decoded color matches the ground truth; wrong-color and invalidly decoded cells count as errors. Agent occupancy and heading are excluded. For state prediction, exact final-cell accuracy requires matching color, identity-aware

agent occupancy, and the heading of any occupying agent. The do-nothing baseline leaves the initial grid and agent states unchanged. Appendix F.1 additionally reports the mode-conditioned checkpoints trained for 160,000 steps with AR, Block-5, Block-15, and bidirectional generation sampled uniformly during training.

C.2.1 Video Prediction

The same horizon and denoising trends persist in video (Table 12). The AR–bidirectional gap at $T = 50$ grows from 0.0094 at 13 total video frames (eight predicted turns) to 0.4364 at 49 total frames (44 predicted turns), and increasing T through 200 does not close it. Therefore, Observations 1 and 4 hold for Turmites in both video and simulator state, while Observation 3 holds in simulator state. Figure 8 visualizes the horizon comparison and denoising-step sweep.

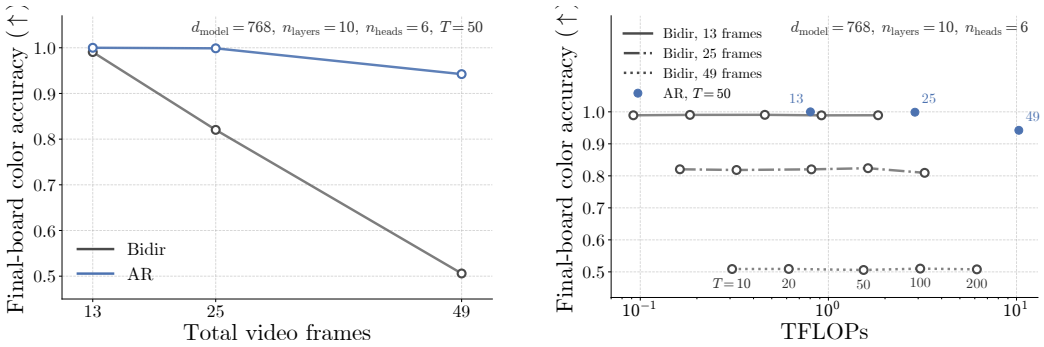


Figure 8: **Turmite video prediction reproduces Observations 1 and 4.** Left: final-board color accuracy at $T = 50$ falls with total video frames for bidirectional generation, while autoregressive accuracy remains high. The 13, 25, and 49 total-frame clips include five conditioning frames and thus have 8, 20, and 44 predicted turns. Right: color accuracy versus estimated TFLOPs per clip, computed as the supplied GFLOPs per clip at $T = 1$ multiplied by $T/1000$. Gray curves show bidirectional results for the three total-frame horizons, with T labeled along the 49-frame curve. Blue points show autoregressive accuracy at $T = 50$; their labels give total frames. Increasing T through 200 does not recover bidirectional accuracy at longer horizons. Values are from Table 12; that table does not report run-to-run ranges.

C.2.2 Simulator-State Prediction

Results In simulator state, autoregressive prediction remains at 1.000 final-grid accuracy throughout the horizon sweep, whereas bidirectional accuracy falls to 0.4964 at 39 turns (Table 13). The do-nothing baseline falls from 0.8100 at four turns to 0.4741 at 39 turns, indicating that later endpoints are harder under this baseline. Figure 9 (left) shows the horizon trend.

More denoising steps do not recover the long-horizon bidirectional model (Table 14). At 39 turns, its accuracy remains approximately 0.496 from $T = 1$ through $T = 50$, while the best autoregressive accuracy is 1.000. Temporal serialization does help: at the same horizon, accuracy improves from 0.496 bidirectionally to 0.749 with Block-15, 0.993 with Block-10, and at least 0.999 with Block-5 or more serial factorizations (Table 15). Figures 9 (right) and 10 visualize these two comparisons.

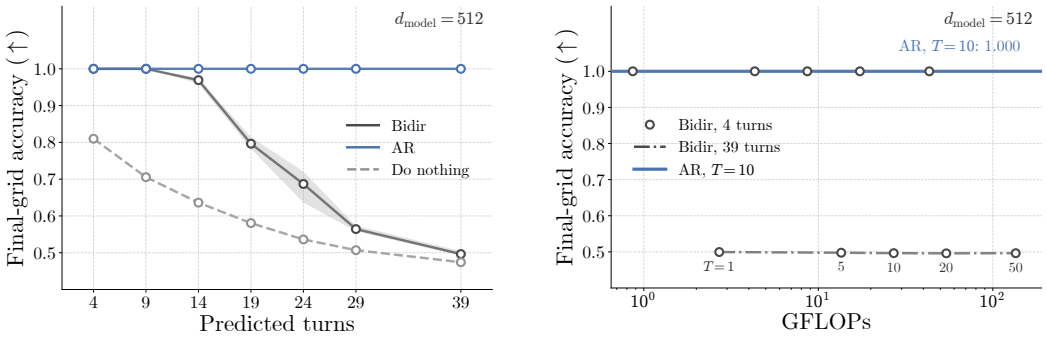


Figure 9: **Turmite simulator-state prediction reproduces Observations 1 and 4.** Left: across predicted turns, autoregressive accuracy remains at 1.000 while bidirectional accuracy falls; shading gives the reported bidirectional range, and the dashed gray curve is the do-nothing baseline. Right: bidirectional accuracy versus estimated GFLOPs per clip. Each point uses its horizon's supplied $T = 1$ GFLOPs multiplied by T; labels along the 39-turn curve give T. Accuracy at 39 turns stays near 0.496 through $T = 50$. The blue reference line marks 1.000 autoregressive accuracy at 39 turns with $T = 10$ in the horizon comparison; the matched raw step-2,000 autoregressive checkpoint also scores 1.000 at $T \in \{1, 5, 10, 20, 50\}$. The line is a horizontal accuracy reference and does not assign an autoregressive compute cost to every plotted abscissa. Accuracy values are from Tables 13 and 14.

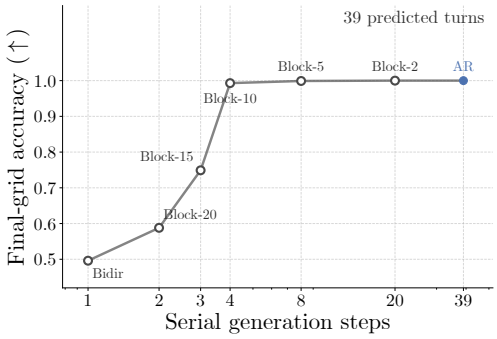


Figure 10: **Temporal factorization in Turmite simulator-state prediction at 39 turns (Observation 3).** Accuracy improves with more serial generation and saturates at Block-2 and autoregression. Values are from Table 15.

C.3 Dependency-Depth Analysis

C.3.1 Hard-Sphere Dynamics

We collapse duplicate records for the same ball pair and time, and group simultaneous contacts sharing a ball into one compound collision node. The collision-dependency graph has one node per resulting event, with edges from preceding events involving any participating ball. Its longest path gives dependency depth D. Equivalently, each ball carries a depth initialized to zero; an isolated collision between balls i, j updates both depths to $1 + \max(d_i, d_j)$, while a compound node updates all participating balls together. Wall bounces are excluded because a ball's isolated motion has a closed-form solution. The total ball–ball collision count C upper-bounds D; the two are equal in only 22.6% of 25-frame trajectories and 3.3% of 49-frame trajectories. Table 16 shows that average measured depth increases with video length alongside the bidirectional–autoregressive gap, supporting f as a proxy for dependency depth in this benchmark.

The relationship also holds within fixed video lengths. At $f = 49$, the bidirectional–autoregressive gap rises from 0.247 for depth 4–7 to 0.278, 0.289, and 0.294 for depths 8–11, 12–15, and 16–19. Fitted depth–gap slopes at $f = 25, 33, 41, 49$ are 0.0026, 0.0053, 0.0034, and 0.0050; a pooled regression with horizon fixed effects gives 0.00426 per dependency step (95% CI $[0.0019, 0.0065]$).

Collision count and depth are strongly correlated in the interacting dataset ($r \approx 0.93$). Stratifying rare trajectories reduces the correlation to about 0.56, but neither variable's coefficient is then statistically separated from zero. The hard-sphere analysis therefore does not identify their separate effects; the fixed-event-count Turmite test below provides a complementary control.

Figure 11 shows the across-horizon and fixed-horizon depth trends.

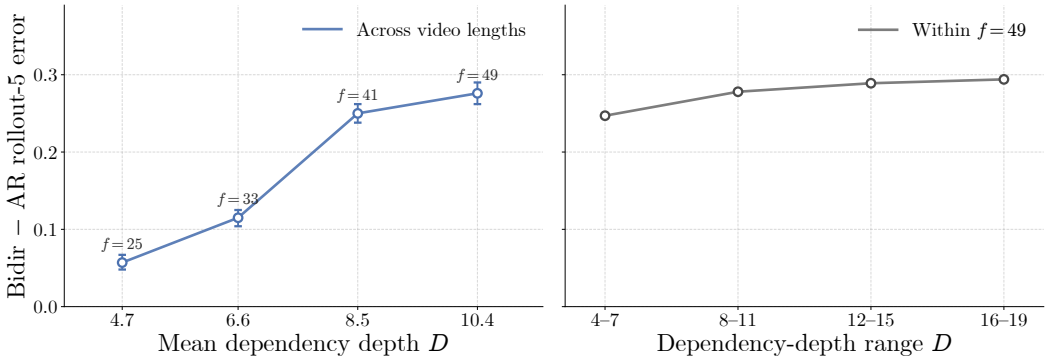


Figure 11: **Dependency depth in hard-sphere video prediction.** Left: the bidirectional–autoregressive rollout-5 error gap against mean dependency depth across video lengths; whiskers are 95% paired cluster-bootstrap confidence intervals. Right: the gap across depth bins at a fixed length of $f = 49$ frames; no intervals were reported for these bins. Across-horizon values are from Table 16.

C.3.2 Two-Agent Turmites at Fixed Event Count

To vary dependency depth while holding event count fixed, we use two nonblocking agents on a shared 5×5 binary grid. They start at distinct positions in the central 3×3 region with independent random headings. The initial state has 25 grid tokens and two agent tokens; each predicted turn adds an active-agent token and a cell-change token. Token type identifies the active agent. For each even total number E of predicted turns, the first agent takes $E/2$ turns, followed by $E/2$ turns by the second. Each event depends on the active agent's preceding state and the latest preceding write to its read cell; an edge between agents arises when the second agent reads a cell modified by the first. The exact longest-path depth d therefore ranges from $E/2$ to E. We construct depth-balanced training and evaluation sets of $20{,}000$ and $1{,}024$ trajectories, respectively, for $E \in \{20, 30, 40, 50\}$.

For each E, we train separate autoregressive and bidirectional state DiTs with $d_{\text{model}} = 512$, four layers, and 16 heads, using the event-token representation and continuous x-prediction objective above. Training uses batch size 256 and learning rate 8×10^{-4}. We evaluate one training seed per

factorization on the same 1,024 held-out trajectories with $T = 10$ Euler denoising steps, using the raw autoregressive step-2,000 checkpoint and raw bidirectional step-40,000 checkpoint. Autoregression denoises and discretizes one turn at a time; bidirectional generation denoises all future tokens jointly.

We measure agent position and heading accuracy across predicted turns, scoring the active agent's row and column jointly and its heading separately. We average turn accuracy within a trajectory and then average trajectories within each exact-depth bin. Unlike final-grid cell accuracy, these measures do not allow repeated flips of the same cell to cancel. To compare horizons, normalize depth as $z = (d - E/2)/(E/2) \in [0, 1]$ and fit bidirectional accuracy as $\hat{a}(z) = \Delta_{\text{depth}} z/100 + C_0$ by equally weighted least squares over depth-bin means. Thus, Δ_{depth} is the fitted change in accuracy, in percentage points, across the full feasible depth range at fixed E. Table 17 also reports the mean autoregressive–bidirectional gap across depths, equally weighting depth bins. Confidence intervals use 10,000 within-bin trajectory bootstraps and reflect evaluation-set uncertainty for one training seed.

Figure 12 summarizes the depth effect and mean gap at each fixed event count.

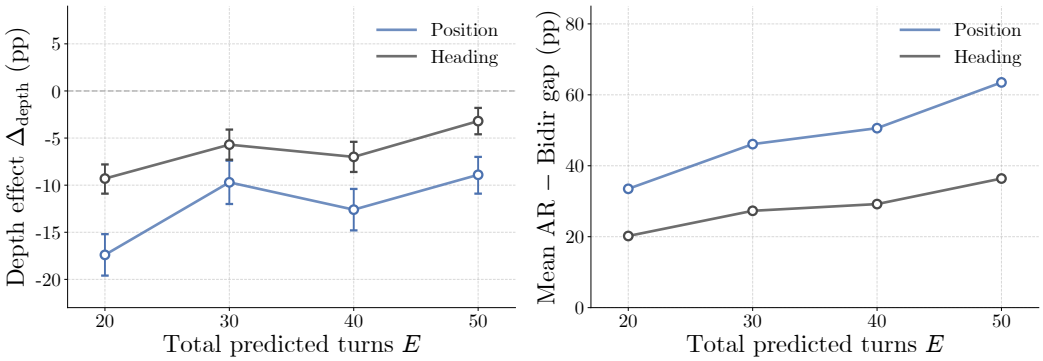
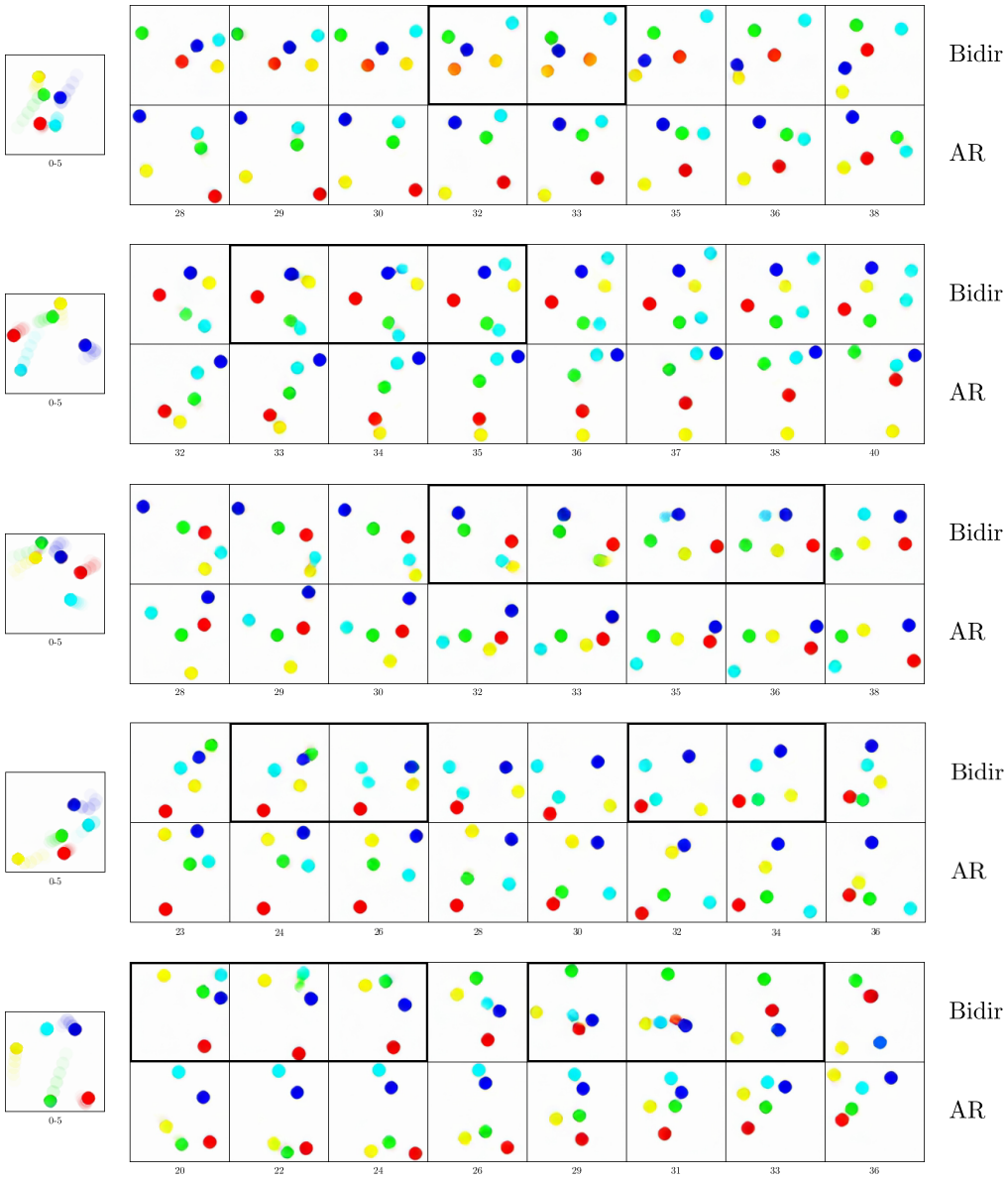


Figure 12: **Dependency depth in two-agent Turmite prediction.** Left: the fitted change in bidirectional position and heading accuracy across the feasible depth range at each fixed event count E; whiskers are 95% within-bin bootstrap confidence intervals. Right: the depth-averaged autoregressive–bidirectional accuracy gap; intervals were not reported for these gaps. Values are from Table 17.

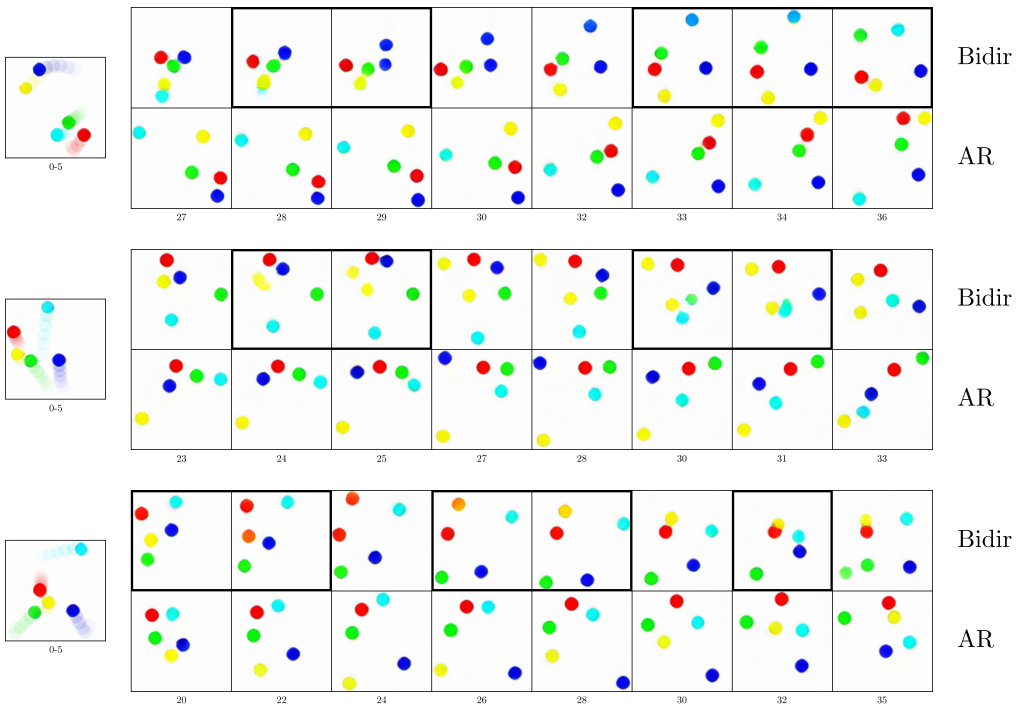
All eight depth-change confidence intervals exclude zero. At each fixed event count, bidirectional position and heading accuracy decrease with dependency depth, while autoregressive accuracy remains near 100% without a detectable depth trend; consequently, the gap widens with depth. The depth-averaged gap also grows with E, although the magnitude of the within-horizon depth effect is not monotonic in E.

D QUALITATIVE RESULTS



(a)

Figure 13: **Qualitative failures.** We compare bidirectional and autoregressive outputs from models with $d_{\text{model}} = 1536$ and $n_{\text{layers}} = 30$. The left column shows the shared conditioning frames with motion trails; the remaining columns show 8 output frames, with frame indices below. Thick borders highlight frames where failures appear. Bidirectional outputs exhibit state-consistency failures such as tunneling, ball-count violations, and color-identity violations, while the autoregressive counterparts remain more physically consistent.



(b)

Figure 13: **Qualitative failures, continued.** Illustrations follow the same format as Figure 13a.

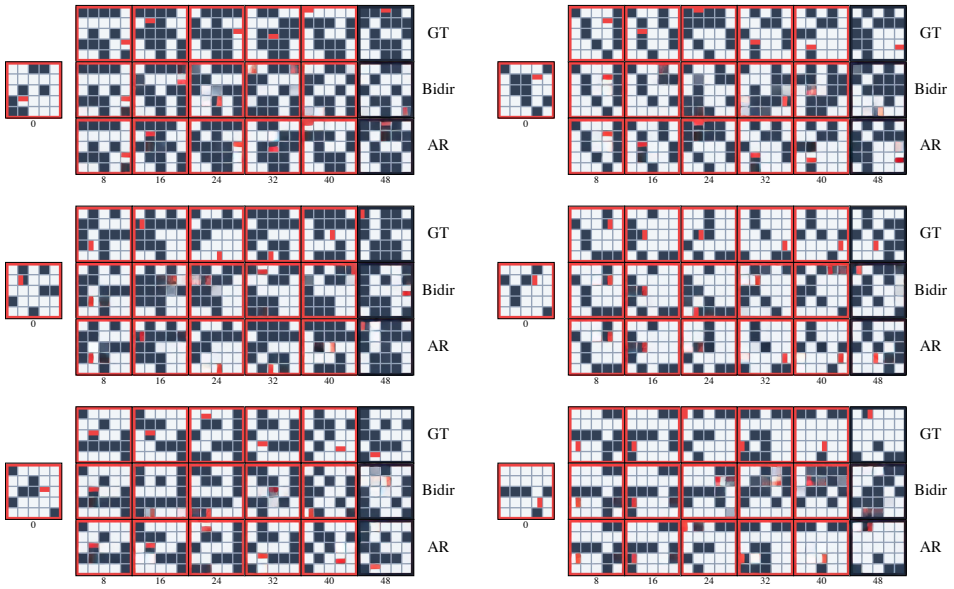


Figure 14: **Turmite video qualitative failures.** Six held-out samples, each showing the first conditioning frame at left and six future frames. Within each sample, the rows show ground truth (GT), bidirectional prediction (Bidir), and autoregressive prediction (AR). The models condition on five frames, although only the first is displayed.

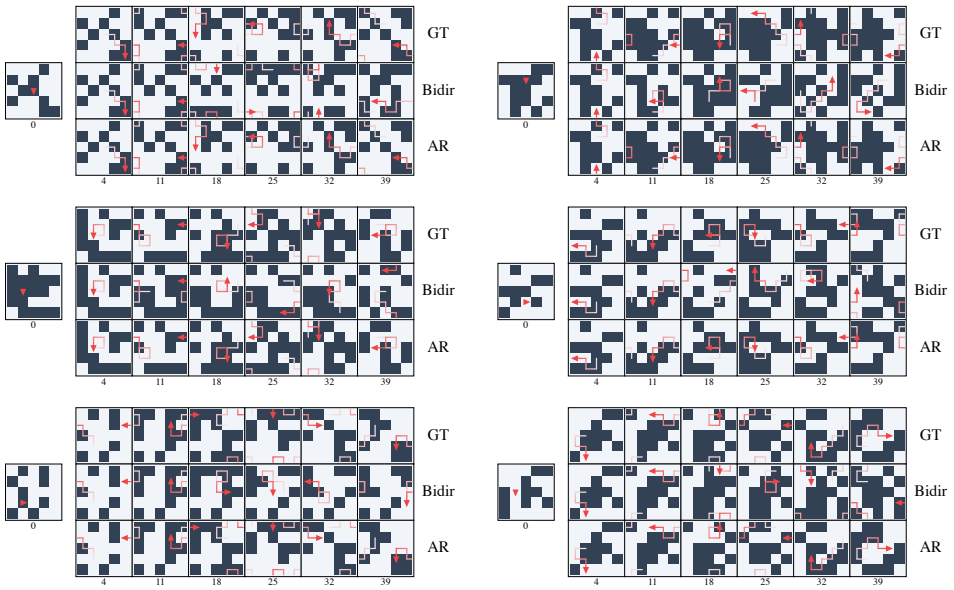


Figure 15: **Single-agent Turmite simulator-state qualitative failures.** Six held-out samples, each showing the initial state and six future steps. Rows show GT, Bidir, and AR.

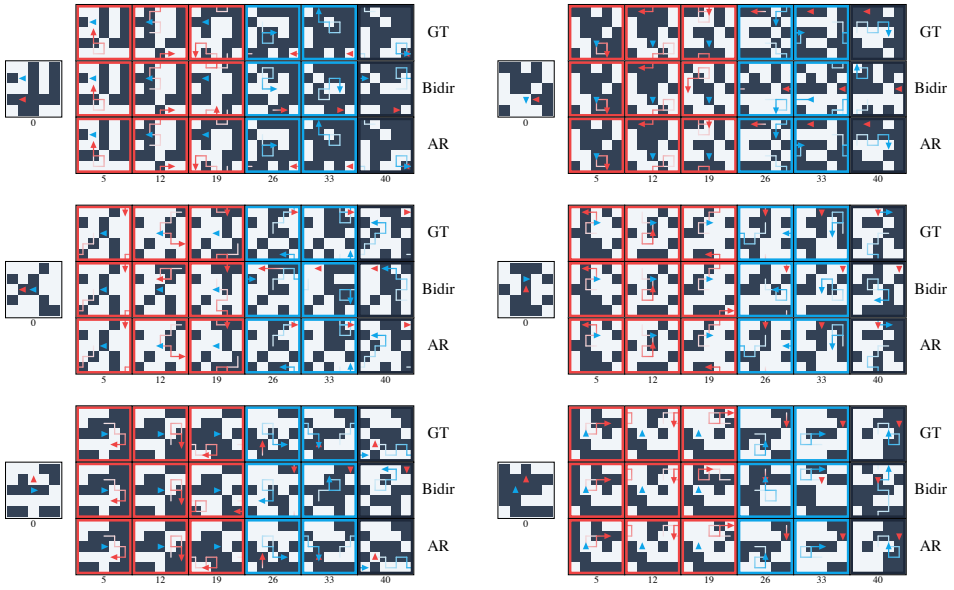


Figure 16: **Two-agent Turmite simulator-state qualitative failures.** Six held-out samples in the same format as Figure 15. Red and blue arrows distinguish the agents; colored panel borders indicate the next active agent.

E VIBRATING-WALL HARD-SPHERE DYNAMICS

In Section 4, we formulate our theory for deterministic video prediction: conditioned on a fully observed initial state, the future trajectory is unique. Natural videos, however, often involve stochasticity, partial observability, or unmodeled perturbations. To test whether the performance gap persists beyond the deterministic setting, we conduct a preliminary experiment on a stochastic variant of the hard-sphere dynamics dataset.

In this vibrating-wall variant, wall collisions are no longer perfectly deterministic. Whenever a ball bounces off a wall, we perturb its speed by a random amount

$$s_{\text{after}} = |s_{\text{before}} + \delta s|, \qquad \delta s \sim \text{Unif}[-6, 6],$$

while keeping the direction of the post-collision velocity unchanged. Thus, even with the same initial state, the future trajectory is no longer unique: different wall vibrations induce different valid rollouts.

The vibrating-wall setting breaks the point-mass assumption used in the deterministic theory, but it does not eliminate all serial structure. Between vibrating-wall collisions, the motion remains deterministic and follows the same local physical laws as in the original hard-sphere system. We therefore evaluate local physical consistency only on deterministic segments. Concretely, we compute the rollout-5 error $\Delta x^{(5)}$ after excluding any frame for which the five-step rollout window contains a ball-to-wall collision. This isolates whether the generated trajectory is locally consistent within intervals where the dynamics are deterministic.

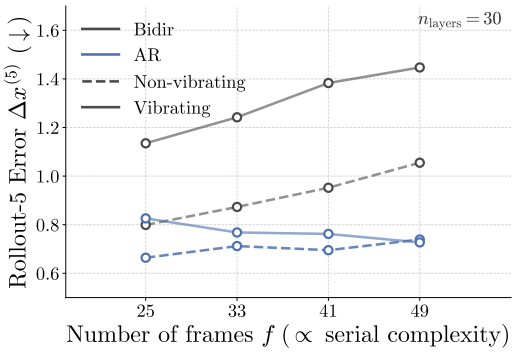


Figure 17: **Preliminary results on vibrating-wall hard-sphere dynamics.** We introduce stochasticity by perturbing the speed of a ball after each wall collision, while leaving the velocity direction unchanged. Evaluation reports rollout-5 error only on deterministic segments, excluding windows that contain a wall collision. Although preliminary, the results show the same qualitative pattern as the deterministic setting, suggesting that the performance gap is not merely an artifact of globally deterministic rollouts.

The results in fig. 17 are preliminary, and we do not claim that this simple vibrating-wall perturbation fully captures the ambiguity of natural video. Nevertheless, the same qualitative trend persists: bidirectional denoising remains worse on local physical consistency than inference schemes with more effective serial computation. This suggests that the deterministic theory identifies one clean mechanism behind the seriality gap, but the phenomenon may extend beyond the exact assumptions of our theoretical result.

F RULING OUT ALTERNATIVE EXPLANATIONS

We test whether the observed advantage of autoregressive generation could be explained by optimization or initialization rather than inference factorization. Specifically, we vary training batch size, repeat training across independent seeds, and fine-tune models initialized from a pretrained Wan checkpoint. In each case, the performance gap persists.

Table 2: **Model trends are consistent under vibrating-wall dynamics.** Physical accuracy and visual quality for bidirectional and autoregressive diffusion models on $n = 5$ vibrating-wall hard-sphere videos of different lengths f.

Type	d_{model}	n_{layers}	n_{params}	TFLOPs	$\Delta x^{(\text{GT})} \downarrow$	$\Delta x^{(1)} \downarrow$	$\Delta x^{(5)} \downarrow$	IoU $\uparrow$
$f = 25$ **frames,** $n = 5$ **balls, vibrating-wall**								
Bidir	768	30	217M	10.4	1.941	0.312	1.135	0.670
AR	768	30	217M	7.3	1.726	0.197	0.826	0.718
$f = 33$ **frames,** $n = 5$ **balls, vibrating-wall**								
Bidir	768	30	217M	13.8	2.454	0.354	1.242	0.660
AR	768	30	217M	10.3	2.281	0.177	0.768	0.728
$f = 41$ **frames,** $n = 5$ **balls, vibrating-wall**								
Bidir	768	30	217M	17.2	2.895	0.397	1.383	0.660
AR	768	30	217M	13.4	2.720	0.177	0.762	0.722
$f = 49$ **frames,** $n = 5$ **balls, vibrating-wall**								
Bidir	768	30	217M	20.9	3.163	0.421	1.447	0.662
AR	768	30	217M	16.6	3.017	0.167	0.727	0.726

F.1 Shared Weights Across Factorizations

For hard-sphere video at $f = 49$, one backbone ($d_{\text{model}} = 768$, $n_{\text{layers}} = 30$) is trained jointly on causal block widths, then evaluated with the same weights, data, and denoising budget under each causal factorization. Its $\Delta x^{(5)}$ decreases from 0.813 (Block-3) to 0.731 (Block-2) to 0.696 (AR). Bidirectional generation was absent from this training mixture, so its out-of-distribution evaluation is not used for the comparison.

For event-token Turmites, each horizon uses one mode-conditioned checkpoint trained by uniformly sampling AR, Block-5, Block-15, or bidirectional generation at each step. The same checkpoint is evaluated under all four modes on $1{,}024$ matched examples with $T = 10$; the reported final-grid cell accuracies are:

F frames	AR	Block-5	Block-15	Bidirectional
20	1.0000	0.9965	0.9244	0.8621
30	1.0000	0.9973	0.7949	0.6286
40	1.0000	0.9989	0.7752	0.5149

Each checkpoint is evaluated after $160{,}000$ training steps (about $40{,}000$ exposures per mode). Bidirectional generation is included in the training mixture, and the ordering holds at all three horizons.

F.2 Batch Size

Diffusion training can be sensitive to batch size, with prior scaling experiments reporting improved generation metrics at larger batch sizes (Li et al., 2024). Because computational constraints limit our main experiments to a batch size $b = 64$, we repeat the autoregressive–bidirectional comparison with $b = 256$. We keep all other settings fixed, including the learning rate of 2×10^{-4}, which we do not retune for the larger batch. The results are shown in Figure 18. Although the $b = 256$ models have higher absolute error, potentially because the learning rate was not retuned, the performance gap persists at both batch sizes.

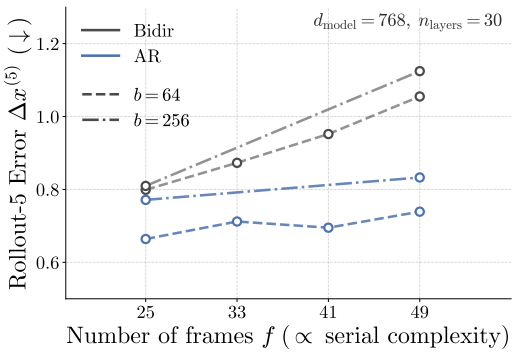


Figure 18: **Batch-size ablation.** Rollout-5 error as a function of video length f for bidirectional and autoregressive models trained with $b \in \{64, 256\}$. The $b = 256$ models have slightly higher error, possibly because the learning rate was not retuned. At both batch sizes, bidirectional error grows more rapidly with f than autoregressive error, preserving the performance gap.

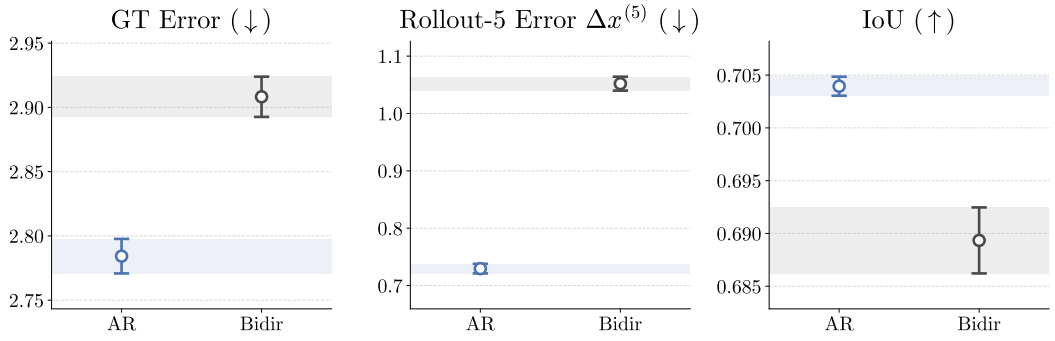


Figure 19: **Robustness to training-seed variation.** Mean global trajectory error $\Delta x^{(\mathrm{GT})}$, rollout-5 error $\Delta x^{(5)}$, and IoU for autoregressive (AR) and bidirectional (Bidir) models across three independent training runs. Error bars and shaded regions denote ± 1 standard deviation. AR has lower errors and higher IoU, with non-overlapping intervals.

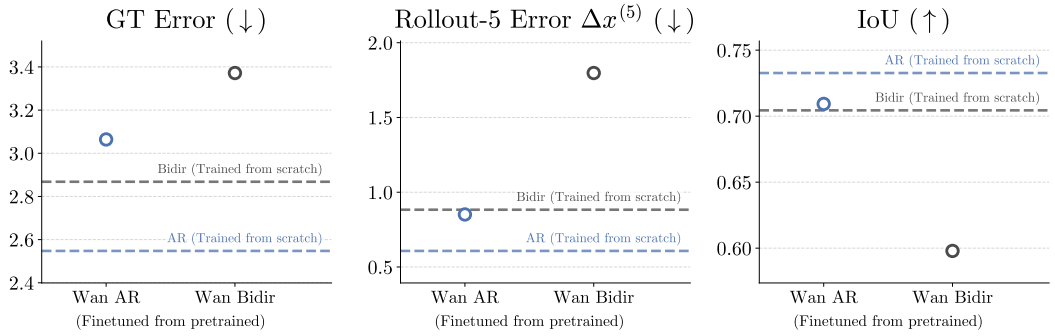


Figure 20: **Wan initialization preserves the model ordering but lowers absolute performance.** Global trajectory error $\Delta x^{(\mathrm{GT})}$, rollout-5 error $\Delta x^{(5)}$, and IoU for models fine-tuned from `Wan2.1-T2V-1.3B`. Circles show the fine-tuned models, and dashed lines show the corresponding models trained from scratch. AR retains lower errors and higher IoU than Bidir, but both fine-tuned models underperform their trained-from-scratch counterparts, possibly because of a resolution mismatch and the out-of-distribution hard-sphere task.

F.3 Training-Seed Variation

Independent diffusion training runs can yield different evaluation outcomes due to parameter initialization, minibatch ordering, and sampled training noise (Dufour et al., 2026). We therefore train each model with three independent seeds at $d_{\text{model}} = 768$ and $n_{\text{layers}} = 30$, using $f = 49$ frames and $n = 5$ balls, and evaluate each run on 1,024 examples. The results are shown in Figure 19. Across all metrics, the error bars are non-overlapping and well separated, indicating that the autoregressive advantage is robust to training-seed variation rather than the result of a favorable seed lottery.

F.4 Fine-Tuning from a Pretrained Checkpoint

To test whether pretraining changes the model ordering, we compare our models trained from scratch with variants initialized from the `Wan-AI/Wan2.1-T2V-1.3B` checkpoint (Wan et al., 2025) and fine-tuned on the hard-sphere dataset. We use the same training configuration, changing only the learning rate from 2×10^{-4} to 2×10^{-5} for fine-tuning. The results are shown in Figure 20.

G Human Evaluation of Perceptual Plausibility

To validate whether our rollout-k error metric reflects perceptual plausibility, we conduct a lightweight human preference study. Evaluators used a blind A/B interface. The interface displays two videos side by side and asks which one looks more physically plausible, with instructions to focus on motion and artifacts rather than sharpness. We compiled judgments from 21 volunteer evaluators. No compensation was provided, and the interface did not request personal or demographic information. Each evaluator rated 7 pairs for each of three comparison groups: AR vs. Bidir, AR vs. Block-3, and Block-3 vs. Bidir, for 21 pairwise judgments per evaluator and 441 total judgments. The evaluators were university-student volunteers, and no prospective IRB or equivalent determination was obtained. Video pairs were drawn from the 128 clips (12.5% of the evaluation set) with the largest rollout-metric separation. This selected-example study is a perceptual sanity check, rather than a measure of preference across randomly sampled trajectories.

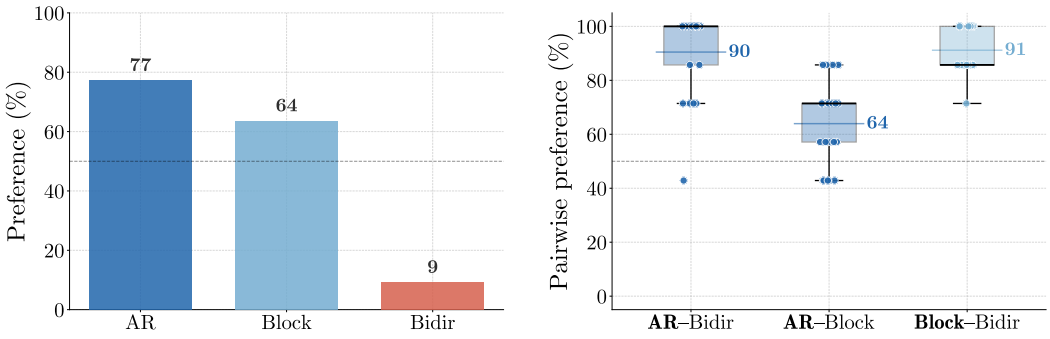


Figure 21: **Human preference study.** Left: aggregate preference rates by model show that AR and Block-3 generations are preferred substantially more often than bidirectional generations. Right: pairwise preferences show strong human preference for AR over Bidir and Block-3 over Bidir, while AR is only moderately preferred over Block-3. This matches the rollout-error ranking, suggesting that large differences in $\Delta x^{(5)}$ correspond to perceptually salient differences in physical plausibility.

The human judgments align with the rollout metric. When $\Delta x^{(5)}$ indicates a large separation, that separation is also perceptible to human evaluators: AR and Block-3 are both strongly preferred over bidirectional generation, whereas the perceptual gap between AR and Block-3 is much smaller. This supports the use of rollout error as a proxy for physically salient generation failures.

H ATTENTION MASKS

Autoregressive and block-autoregressive variants use the same base architecture but replace full temporal attention with FlexAttention (Dong et al., 2024) block-causal masks over temporal latent frames.

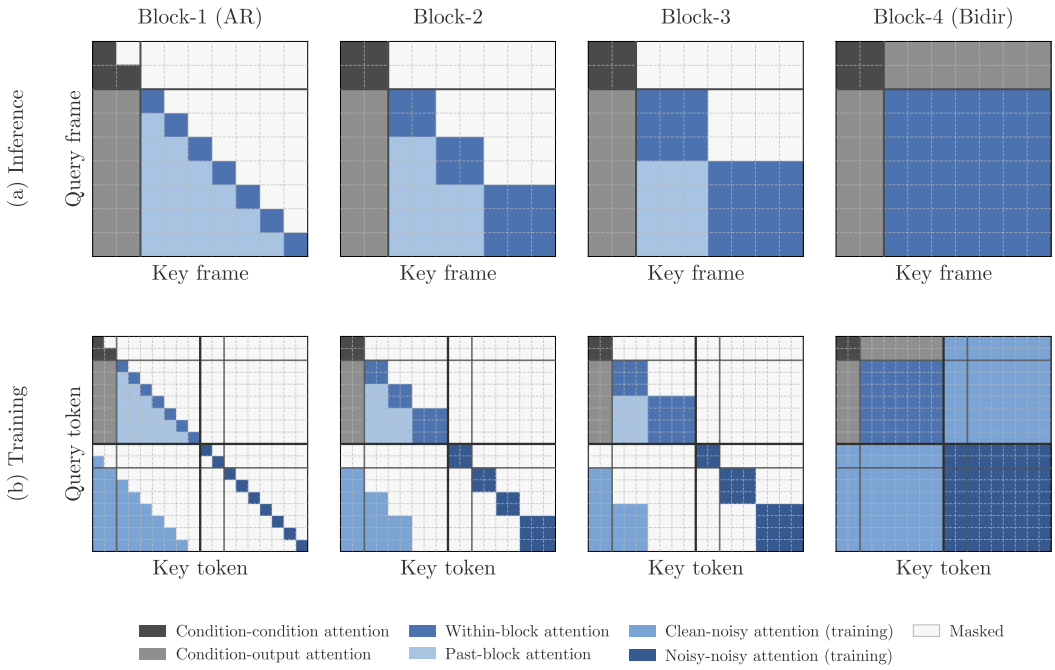


Figure 22: **Attention masks for blockwise generation.** The masks are illustrated for the $f = 25$ setting. Rows denote query positions and columns denote key/value positions; colored cells indicate permitted attention entries and white cells are masked. The top row shows inference-time masks over condition and output frames, while the bottom row shows the corresponding training-time masks over clean and noisy token copies. Columns compare autoregressive Block-1 masking, intermediate block-causal masks, and bidirectional masking. Thick lines mark condition/output and clean/noisy boundaries; dashed lines show individual frame or token cells.

I NUMERICAL RESULTS

Table 3: **Results under different video lengths and inference factorizations.** Physical accuracy and visual fidelity for bidirectional, blockwise, and autoregressive diffusion models on $n = 5$ hard-sphere dynamics videos spanning a range of lengths f. Lower $\Delta x^{(\mathrm{GT})}$, $\Delta x^{(1)}$, and $\Delta x^{(5)}$ indicate higher physical accuracy; higher IoU indicates higher visual fidelity. The results support Observations 1 and 3: bidirectional models degrade with longer chains of dependent events, while temporal serialization improves local physical accuracy.

Type	d_{model}	n_{layers}	n_{params}	TFLOPs	$\Delta x^{(\mathrm{GT})} \downarrow$	$\Delta x^{(1)} \downarrow$	$\Delta x^{(5)} \downarrow$	IoU $\uparrow$
$f = 25$ **frames,** $n = 5$ **balls**								
Bidir	768	30	217M	10.4	1.441	0.197	0.799	0.703
Bidir	1536	30	867M	39.9	1.265	0.160	0.670	0.726
AR	768	30	217M	7.3	1.179	0.166	0.664	0.714
AR	1536	30	867M	28.1	1.042	0.148	0.613	0.730
$f = 33$ **frames,** $n = 5$ **balls**								
Bidir	768	30	217M	13.8	2.047	0.210	0.873	0.693
Bidir	1536	30	867M	52.0	1.944	0.164	0.718	0.725
AR	768	30	217M	10.3	1.905	0.175	0.712	0.705
AR	1536	30	867M	39.6	1.645	0.142	0.604	0.735
$f = 41$ **frames,** $n = 5$ **balls**								
Bidir	768	30	217M	17.2	2.581	0.227	0.952	0.696
Bidir	1536	30	867M	64.4	2.402	0.200	0.856	0.698
AR	768	30	217M	13.4	2.423	0.167	0.695	0.712
AR	1536	30	867M	51.3	2.165	0.142	0.606	0.732
$f = 49$ **frames,** $n = 5$ **balls**								
Bidir	768	30	217M	20.9	2.923	0.261	1.055	0.691
Bidir	1536	30	867M	77.0	2.868	0.205	0.883	0.704
Block-3	768	30	217M	16.9	2.749	0.182	0.762	0.713
Block-3	1536	30	867M	63.7	2.675	0.174	0.716	0.705
Block-2	768	30	217M	16.7	2.717	0.169	0.708	0.715
Block-2	1536	30	867M	63.4	2.578	0.145	0.620	0.731
AR	768	30	217M	16.6	2.798	0.178	0.739	0.703
AR	1536	30	867M	63.1	2.548	0.142	0.607	0.733

Table 4: **Uncertainty in hard-sphere video rollout-5 error.** Each row compares autoregressive (AR) and bidirectional (Bidir) generation on the same $1{,}024$ five-ball evaluation trajectories. The AR and Bidir columns give 95% confidence intervals for their respective mean $\Delta x^{(5)}$ errors. The gap column gives Bidir minus AR with a 95% paired-bootstrap confidence interval from $100{,}000$ trajectory resamples. Gaps were computed before rounding the displayed means.

d_{model}	Frames f	AR [95% CI]	Bidir [95% CI]	Gap [95% paired CI]
768	25	0.664 [0.653, 0.675]	0.799 [0.786, 0.812]	0.135 [0.124, 0.147]
768	33	0.712 [0.701, 0.724]	0.873 [0.859, 0.886]	0.160 [0.147, 0.174]
768	41	0.695 [0.687, 0.704]	0.952 [0.939, 0.966]	0.257 [0.243, 0.270]
768	49	0.739 [0.730, 0.748]	1.055 [1.038, 1.072]	0.316 [0.299, 0.334]
1536	25	0.613 [0.602, 0.624]	0.670 [0.659, 0.682]	0.057 [0.048, 0.067]
1536	33	0.604 [0.594, 0.613]	0.718 [0.707, 0.730]	0.115 [0.104, 0.125]
1536	41	0.606 [0.598, 0.615]	0.856 [0.844, 0.868]	0.250 [0.238, 0.262]
1536	49	0.607 [0.599, 0.616]	0.883 [0.870, 0.897]	0.276 [0.262, 0.290]

Table 5: **Extractor sensitivity: mean rollout-5 error.** Mean $\Delta x^{(5)}$ (lower is better) for 49-frame hard-sphere videos over 1,024 trajectories per model. Each width is evaluated with eight extractors; centroid ± 32 is the published setting.

d_{model}	Extractor	AR	Block-2	Block-3	Bidir
768	Centroid ± 8	2.2603	2.2580	2.2307	2.3008
768	Centroid ± 16	1.5683	1.5298	1.5361	1.6936
768	Centroid ± 32	0.7386	0.7081	0.7623	1.0551
768	Centroid ± 64	0.4690	0.4554	0.5416	0.8851
768	Centroid ± 96	0.4478	0.4368	0.5281	0.8742
768	Median ± 32	0.8386	0.8033	0.8529	1.1284
768	Largest component ± 32	0.7929	0.7583	0.8092	1.0990
768	Nearest palette	0.5553	0.5371	0.6159	0.9427
1536	Centroid ± 8	2.2642	2.2539	2.2670	2.2788
1536	Centroid ± 16	1.4734	1.4780	1.5798	1.6136
1536	Centroid ± 32	0.6074	0.6201	0.7159	0.8830
1536	Centroid ± 64	0.3232	0.3533	0.4139	0.6901
1536	Centroid ± 96	0.2970	0.3308	0.3881	0.6789
1536	Median ± 32	0.7205	0.7289	0.8166	0.9694
1536	Largest component ± 32	0.6625	0.6731	0.7723	0.9330
1536	Nearest palette	0.4101	0.4359	0.4945	0.7633

Table 6: **Extractor sensitivity: paired rollout-5 error gaps.** Model-minus-AR mean differences with 95% paired trajectory-bootstrap percentile intervals from 10,000 resamples of the same 1,024 trajectories per model (RNG seed 0). Positive gaps indicate higher error than AR. These saved comparisons do not include direct Block-3-minus-Block-2 or Bidir-minus-Block-3 intervals.

d_{model}	Extractor	Block-2 − AR [95% CI]	Block-3 − AR [95% CI]	Bidir − AR [95% CI]
768	Centroid ± 8	-0.0023 [-0.0266, 0.0211]	-0.0296 [-0.0531, -0.0059]	0.0406 [0.0152, 0.0654]
768	Centroid ± 16	-0.0385 [-0.0541, -0.0226]	-0.0322 [-0.0473, -0.0171]	0.1253 [0.1073, 0.1440]
768	Centroid ± 32	-0.0305 [-0.0408, -0.0199]	0.0238 [0.0133, 0.0344]	0.3165 [0.2992, 0.3340]
768	Centroid ± 64	-0.0136 [-0.0211, -0.0061]	0.0726 [0.0637, 0.0816]	0.4161 [0.3993, 0.4329]
768	Centroid ± 96	-0.0110 [-0.0188, -0.0033]	0.0803 [0.0704, 0.0905]	0.4264 [0.4093, 0.4438]
768	Median ± 32	-0.0352 [-0.0457, -0.0251]	0.0143 [0.0044, 0.0249]	0.2898 [0.2723, 0.3072]
768	Largest component ± 32	-0.0346 [-0.0454, -0.0238]	0.0163 [0.0057, 0.0272]	0.3061 [0.2879, 0.3242]
768	Nearest palette	-0.0181 [-0.0266, -0.0097]	0.0606 [0.0504, 0.0708]	0.3874 [0.3717, 0.4039]
1536	Centroid ± 8	-0.0103 [-0.0321, 0.0121]	0.0028 [-0.0226, 0.0276]	0.0146 [-0.0103, 0.0393]
1536	Centroid ± 16	0.0046 [-0.0107, 0.0195]	0.1063 [0.0910, 0.1215]	0.1402 [0.1243, 0.1562]
1536	Centroid ± 32	0.0128 [0.0036, 0.0221]	0.1085 [0.0985, 0.1187]	0.2756 [0.2614, 0.2897]
1536	Centroid ± 64	0.0301 [0.0242, 0.0361]	0.0907 [0.0834, 0.0981]	0.3669 [0.3541, 0.3799]
1536	Centroid ± 96	0.0338 [0.0277, 0.0401]	0.0912 [0.0838, 0.0985]	0.3819 [0.3686, 0.3949]
1536	Median ± 32	0.0084 [-0.0007, 0.0174]	0.0961 [0.0859, 0.1062]	0.2489 [0.2346, 0.2632]
1536	Largest component ± 32	0.0105 [0.0010, 0.0202]	0.1098 [0.0994, 0.1201]	0.2705 [0.2554, 0.2852]
1536	Nearest palette	0.0258 [0.0185, 0.0331]	0.0844 [0.0757, 0.0930]	0.3532 [0.3401, 0.3666]

Table 7: **Single-ball control as a non-serial setting.** Physical accuracy and visual fidelity for bidirectional and autoregressive diffusion models on $n = 1$ hard-sphere dynamics videos of different lengths f. Unlike the multi-ball setting, longer single-ball videos do not introduce longer chains of dependent events. The results support Observation 2: the bidirectional–autoregressive gap largely disappears without serial dependence.

Type	d_{model}	n_{layers}	n_{params}	TFLOPs	$\Delta x^{(\text{GT})} \downarrow$ $n = 1$	$\Delta x^{(5)} \downarrow$ $n = 1$	$\Delta x^{(5)} \downarrow$ $n = 5$	IoU $\uparrow$ $n = 1$
$f = 25$ **frames**								
Bidir	768	30	217M	10.4	0.310	0.220	0.799	0.823
AR	768	30	217M	7.3	0.162	0.201	0.664	0.819
$f = 49$ **frames**								
Bidir	768	30	217M	20.9	1.417	0.246	1.055	0.810
AR	768	30	217M	16.6	0.471	0.220	0.739	0.815

Five-ball separate-lane control We confine each of five balls to its own lane, removing ball–ball collisions while retaining object count and the $f \in \{25, 49\}$ prediction horizons. Matched autoregressive and bidirectional models ($d_{\text{model}} = 768$, $n_{\text{layers}} = 30$) use the same $\Delta x^{(5)}$ evaluation. At $f = 25$, their errors are 0.343 and 0.344, respectively (bidirectional minus autoregressive: $+0.001$); at $f = 49$, they are 0.365 and 0.360 (-0.005). By comparison, the interacting five-ball gaps are $+0.135$ and $+0.316$ at the same horizons.

Table 8: **More denoising steps do not close the performance gap.** Local physical accuracy for bidirectional diffusion models with $T \in \{10, 20, 50, 100, 200\}$ denoising steps on $n = 5$ hard-sphere dynamics videos of different lengths f. Performance plateaus after a moderate number of steps, while longer videos remain more difficult. The results support Observation 4: denoising depth does not provide scalable serial computation for longer chains of dependent events.

Type	d_{model}	n_{layers}	TFLOPs	$\Delta x^{(5)} \downarrow$ $T = 10$	$\Delta x^{(5)} \downarrow$ $T = 20$	$\Delta x^{(5)} \downarrow$ $T = 50$	$\Delta x^{(5)} \downarrow$ $T = 100$	$\Delta x^{(5)} \downarrow$ $T = 200$
$f = 25$ **frames,** $n = 5$ **balls**								
Bidir	1536	30	39.9	0.690	0.667	0.670	0.679	0.689
$f = 33$ **frames,** $n = 5$ **balls**								
Bidir	1536	30	52.0	0.746	0.719	0.718	0.724	0.740
$f = 41$ **frames,** $n = 5$ **balls**								
Bidir	1536	30	64.4	0.946	0.886	0.856	0.847	0.848
$f = 49$ **frames,** $n = 5$ **balls**								
Bidir	1536	30	77.0	0.991	0.931	0.883	0.889	0.916

Table 9: **Depth scaling is more effective than width scaling.** Physical accuracy and visual fidelity for bidirectional diffusion models with different widths d_{model} and depths n_{layers} on $n = 5$ hard-sphere dynamics videos. Depth improves physical accuracy more strongly than width up to about 30 layers, beyond which further improvements are unclear. The results support Observation 5: useful serial computation is supplied more effectively by backbone depth than by width alone.

Type	d_{model}	n_{layers}	n_{params}	TFLOPs	$\Delta x^{(\text{GT})} \downarrow$	$\Delta x^{(1)} \downarrow$	$\Delta x^{(5)} \downarrow$	IoU $\uparrow$
$f = 49$ **frames,** $n = 5$ **balls**								
Bidir	640	30	151M	14.9	2.973	0.307	1.172	0.665
Bidir	768	20	146M	14.0	3.038	0.403	1.420	0.640
Bidir	768	30	217M	20.9	2.923	0.261	1.055	0.691
Bidir	768	60	430M	41.7	2.839	0.233	0.950	0.696
Bidir	768	90	643M	62.6	2.788	0.210	0.881	0.700
Bidir	1152	10	169M	14.8	3.076	0.416	1.466	0.653
Bidir	1152	20	329M	29.7	2.941	0.261	1.060	0.690
Bidir	1152	30	488M	44.5	2.822	0.227	0.951	0.694
Bidir	1152	50	807M	74.2	2.739	0.251	1.000	0.676
Bidir	1536	10	301M	25.7	2.939	0.324	1.230	0.680
Bidir	1536	30	867M	77.0	2.868	0.205	0.883	0.704

Table 10: **Direct hard-sphere simulator-state prediction across horizons.** Rollout-5 drift rate (m/s), equal to mean rollout displacement divided by $5/15\,\text{s}$, at $T = 20$ from step-40,000 EMA checkpoints. Brackets give the minimum and maximum over three training seeds.

Frames f	Bidir [min, max]	AR [min, max]	Bidir − AR
8	0.221 [0.195, 0.239]	0.210 [0.208, 0.214]	0.011
16	1.439 [1.364, 1.488]	0.803 [0.772, 0.819]	0.636
24	2.297 [2.214, 2.404]	0.874 [0.860, 0.894]	1.423
32	2.587 [2.521, 2.690]	0.898 [0.874, 0.930]	1.689
40	2.590 [2.511, 2.639]	0.829 [0.795, 0.850]	1.761
48	2.738 [2.659, 2.823]	0.782 [0.753, 0.817]	1.956

Table 11: **Denoising-step sweep for direct simulator-state prediction.** Each entry gives the reported AR / Bidir rollout-5 drift rate (m/s) for averaged step-40,000 checkpoints on 256 held-out trajectories.

f	$T = 1$	$T = 2$	$T = 5$	$T = 10$	$T = 20$	$T = 50$
8	.22730 / .26792	.21830 / .23934	.21087 / .23431	.20478 / .23580	.20699 / .23278	.21260 / .23923
16	.89817 / 1.38505	.90763 / 1.38642	.86279 / 1.38636	.81571 / 1.40524	.81730 / 1.46740	.81911 / 1.55872
24	.97356 / 2.10910	1.00201 / 2.10314	.93399 / 2.14198	.87741 / 2.19116	.89977 / 2.27589	.85386 / 2.42231
32	.99521 / 2.37300	1.00584 / 2.36259	.95465 / 2.44243	.87294 / 2.49307	.87979 / 2.61708	.84381 / 2.73050
40	.89058 / 2.48341	1.00143 / 2.40063	.91048 / 2.49633	.84120 / 2.55401	.83344 / 2.64492	.89767 / 2.80719
48	.94029 / 2.49991	.92806 / 2.32006	.88097 / 2.49155	.82484 / 2.58051	.81036 / 2.65584	.83233 / 2.75436

Table 12: **Turmite video prediction.** Horizons are total video frames, including five conditioning frames: $13/25/49$ frames give $8/20/44$ predicted turns. Each denoising-budget entry gives AR / bidirectional final-board color accuracy. Gap@50 is the difference between the displayed AR and bidirectional accuracies at $T = 50$.

Total frames	Gap@50	$T = 10$	$T = 20$	$T = 50$	$T = 100$	$T = 200$
13	.0094	.9996 / .9892	.9995 / .9904	1.0000 / .9906	.9995 / .9890	.9996 / .9893
25	.1785	.9968 / .8205	.9987 / .8183	.9988 / .8203	.9976 / .8239	.9947 / .8094
49	.4364	.9332 / .5089	.9510 / .5092	.9423 / .5059	.9322 / .5102	.8676 / .5079

Table 13: **Turmite simulator-state prediction across horizons.** Exact final-grid cell-level accuracy. Brackets reproduce the ranges reported for bidirectional prediction.

Predicted turns	Bidir [reported range]	AR	Do nothing
4	1.0000 [1.0000, 1.0000]	1.0000	0.8100
9	1.0000 [1.0000, 1.0000]	1.0000	0.7054
14	0.9692 [0.9599, 0.9746]	1.0000	0.6362
19	0.7964 [0.7824, 0.8144]	1.0000	0.5806
24	0.6869 [0.6364, 0.7205]	1.0000	0.5361
29	0.5643 [0.5585, 0.5734]	1.0000	0.5070
39	0.4964 [0.4921, 0.5046]	1.0000	0.4741

Table 14: **Denoising-step sweep for Turmite simulator-state prediction.** Bidirectional final-grid accuracy and its gap to the best autoregressive score (1.0000) at short and long horizons.

	4 turns		39 turns	
T	Bidir	Gap to AR	Bidir	Gap to AR
1	1.0000	0.0000	0.4995	0.5005
5	1.0000	0.0000	0.4976	0.5024
10	1.0000	0.0000	0.4964	0.5036
20	1.0000	0.0000	0.4959	0.5041
50	1.0000	0.0000	0.4963	0.5037

Table 15: **Turmite simulator-state factorization ladder at 39 turns.** More serial factorizations approach autoregressive accuracy.

Factorization	Turns per step	Serial steps	Accuracy [reported range]
AR (Block-1)	1	39	1.000 [1.000, 1.000]
Block-2	2	20	1.000 [1.000, 1.000]
Block-5	5	8	0.999 [0.998, 1.000]
Block-10	10	4	0.993 [0.988, 0.995]
Block-15	15	3	0.749 [0.670, 0.792]
Block-20	20	2	0.588 [0.581, 0.597]
Bidirectional	39	1	0.496 [0.492, 0.505]

Table 16: **Hard-sphere dependency depth across video lengths.** Mean ball–ball collision count C, mean dependency depth D, and rollout-5 error for $d_{\text{model}} = 1536$, $n_{\text{layers}} = 30$. Gap intervals are 95% paired cluster-bootstrap confidence intervals.

Frames f	Mean C	Mean D	AR	Bidir	Bidir − AR [95% CI]
25	5.99	4.72	0.613	0.670	0.057 [0.048, 0.067]
33	8.44	6.56	0.604	0.718	0.115 [0.104, 0.125]
41	10.99	8.46	0.606	0.856	0.250 [0.238, 0.262]
49	13.49	10.36	0.607	0.883	0.276 [0.262, 0.290]

Table 17: **Two-agent Turmites at fixed event count.** Bidirectional position and heading accuracy decrease with exact dependency depth at every tested E; autoregressive accuracy remains near 100% without a detectable depth trend. Changes and mean AR–bidirectional gaps are in percentage points.

E (depth range)	Position Δ_{depth} [95% CI]	Position gap	Heading Δ_{depth} [95% CI]	Heading gap
20 (10–20)	-17.4 [-19.6, -15.2]	33.5	-9.3 [-10.9, -7.8]	20.2
30 (15–30)	-9.7 [-12.0, -7.4]	46.1	-5.7 [-7.3, -4.1]	27.3
40 (20–40)	-12.6 [-14.8, -10.4]	50.6	-7.0 [-8.6, -5.4]	29.2
50 (25–50)	-8.9 [-10.9, -7.0]	63.5	-3.2 [-4.6, -1.8]	36.4