PROBING KINEMATIC CHAIN REASONING IN VIDEO DIFFUSION TRANSFORMERS: A STUDY OF 2D GEAR SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Video diffusion transformers (DiTs) are increasingly viewed as promising models for learning physical dynamics directly from video. However, off-the-shelf models struggle to generate physically plausible gear motions, even in simple scenarios involving only two gears. Simulating gear mechanisms is conceptually simple yet challenging, as the motion of a single gear strictly dictates the kinematics of the entire system. In particular, determining the rotation direction of each gear requires computing its rotational parity by traversing the underlying kinematic chain. To investigate whether video DiTs can learn such kinematic chain reasoning, we utilize 2D involute gear trains as a testbed to train and analyze video DiTs. Our analysis reveals that models can indeed learn kinematic chain reasoning, but they acquire two distinct types of reasoning mechanisms depending on the kinematic heights encountered during training: when trained on mechanisms with short kinematic heights, the model acquires a parallel BFS-like reasoning, using transformer layers as breadth-first search steps to incrementally determine parity across the kinematic tree. Conversely, when exposed to large kinematic heights during training, the model adopts a divide-and-conquer-like strategy—first resolving parity within local neighborhoods and subsequently merging them to achieve global consistency. Overall, this work demonstrates that video DiTs are capable of learning algorithmic reasoning over kinematic chains, while also uncovering their generalization limits and highlighting the critical role of training data complexities. *Anonymous Project Page:* `https://Cog-DiT.github.io`

1 INTRODUCTION

Spatial intelligence, intended as the ability to solve problems in the physical world, is one of the most active frontiers of artificial intelligence (AI). While many of the recent advances in AI are based on language generation, spatial intelligence challenges models that are exclusively, or even primarily, trained on text. Text is in fact a low-bandwidth modality which makes it difficult to capture the geometry, physical properties, and affordances of the physical world. Hence, many authors have considered image-centric models as a better basis for spatial intelligence. They have suggested, in particular, that video generators can be a means to learn world models, i.e., to predict the consequences of actions in the world, to reason about physical phenomena, and to solve physical problems (Wiedemer et al., 2025b; Tong et al., 2026).

Video models have several appealing features, starting from their ability to learn from vast quantities of unlabelled Internet videos. Even so, these models are usually optimised for the perceptual quality of their samples, and not for the accuracy or physical plausibility of their predictions, or for their ability to solve physically-grounded problems. While some authors have tested the ability of video generators to model physical systems (Bansal et al., 2025; Sun et al., 2025; Bansal et al., 2026; Motamed et al., 2025; Chen et al., 2025; Zhang et al., 2025a; Guo et al., 2026; Gu et al., 2026; Lin et al., 2025), these explorations have been limited to small systems comprising only a handful of interacting objects. In this paper, we ask whether these capabilities extend to much more complex systems consisting of dozens of simultaneously interacting parts.

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.

We propose to consider systems of 2D involute gears as a representative testbed. Gear systems are conceptually simple, but predicting their motion requires reasoning over potentially long kinematic chains: once the motion of a single gear is specified, the rotational parity (e.g., clockwise vs. counter-clockwise) of every other gear is determined by the chain of kinematic interactions. Specifically, each gear rotates in the opposite direction to its immediate neighbor, so gears two hops apart rotate in the same direction, and so on. Thus, to generate a physically plausible animations, the video model must correctly resolve these kinematic chain dependencies simultaneously. Indeed, as demonstrated in Fig. 2, we observe that commercial models (Sharma et al., 2024; Luma AI, 2026) fail to produce physically plausible gear interactions, sometimes even in trivial scenarios involving only two gears.

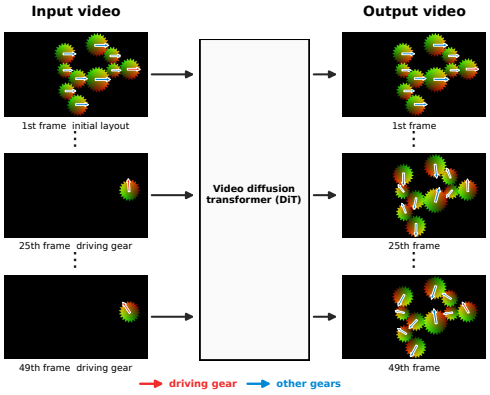


Figure 1: **Task definition.** Given an image specifying the the spatial layout of a gear system as the first frame and the motion of a single driving gear as a video, the video diffusion transformer is tasked with generating the fully animated gear system while satifying the implied kinematic constraints. **See the webpage for videos.**

Motivated by these failures, we fine-tune an open-source video diffusion transformer (DiT) (Wan, 2025) on our synthetic animated gear datasets. As illustrated in Fig. 1, we condition the model on the complete spatial layout of the gear system and the motion of a single driving gear, and task it with generating the fully animated gear system. We quantitatively evaluate the motion accuracy of individual generated gears using our evaluation pipeline, and further develop feature-probing techniques to analyze the internal reasoning mechanisms that emarge in the trained video DiTs.

Our experiments reveal that video DiTs learn two distinct types of reasoning mechanisms depending on the *kinematic height* encountered during training, defined as the maximum graph distance between the driving gear and any other gear. Specifically, when exposed only to relatively short kinematic heights (i.e., 5 and 10), the models learn a *parallel BFS*-like reasoning mechanism. Starting from the driving gear, the models gradually determine rotational parity by propagating information to immediate neighbors in parallel, with successive steps implemented by successive transformer layers in the DiT. With this mechanism, the models exhibit strong zero-shot generalization to unseen larger gear counts, provided that the kinematic height remains within the training distribution. However, the models fail to generalize to unseen kinematic heights, as they learn to execute only as many BFS steps as are minimally necessary to cover the kinematic heights encountered during training.

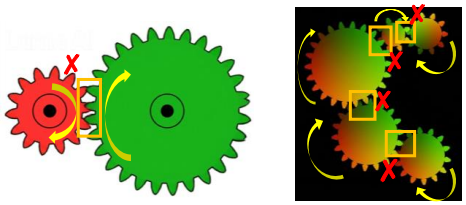


Figure 2: **Animating gear mechanisms with commercial video generative models.** Veo3.1 (Google DeepMind, 2026) and Ray 3.14 (Luma AI, 2026) often violate basic kinematic constraints, producing conflicting rotational directions between meshing gears. Yellow arrows indicate gear rotation. **See the supplemental material for videos.**

When trained on larger kinematic heights, however, the models adopt a different strategy, as the number of available transformer depth is insufficient to carry out BFS steps to the required depth. Specifically, while the models continue to perform parallel BFS-like propagation, they also begin to determine locally consistent gear parities in parallel in regions that are unreachable by the BFS from the driving gear, and subsequently merge these local parities to make them globally consistent. The resulting divide-and-conquer-like mechanism allows the models to resolve long-chain kinematic dependencies efficiently within the limited number of transformer layers available. Although this strategy enables generalization to some unseen kinematic heights, it does not generalize universally and can fail even on unseen topologies whose kinematic heights are within the training distribution.

Overall, our experiments reveal intriguing properties of video DiTs applied to kinematic chain reasoning tasks. In particular, while video DiTs *can* be taught to perform variable-length kinematic chain reasoning, the fact that they learn distinct reasoning mechanisms depending on the topological complexity of the training data distribution demonstrates that video DiTs have limited generalizability to out-of-distribution problem complexity. We hope our findings guide future work toward training strategies and novel architectures with more powerful physical reasoning capabilities.

In summary, our contributions include:

- **A testbed and evaluation pipeline for long-chain kinematic reasoning:** We introduce a synthetic dataset of 2D involute gear systems, together with an evaluation pipeline and feature-probing methodology, enabling systematic analysis of kinematic chain reasoning capabilities of video diffusion transformers (DiTs).
- **Discovery of a parallel BFS-like reasoning mechanism in short kinematic chains:** We show that video DiTs trained on short kinematic chains learn a parallel breadth-first search (BFS) mechanism that determines rotational parity layer by layer. We demonstrate that this mechanism enables zero-shot generalization to larger gear counts provided that the kinematic height remains within the training bound.
- **Discovery of divide-and-conquer-like reasoning in long chains:** We demonstrate that when parallel BFS requires more layers than are available in the DiT, models transition to a divide-and-conquer-like mechanism. They form locally consistent parity blocks in parallel and subsequently merge these blocks globally, enabling long-chain kinematic reasoning with a limited number of layers.

2 Related work

Physically plausible video generation. Recent progress in video generative models (Wan, 2025; Kong et al., 2025; Genmo Team, 2024; Luma AI, 2025; NVIDIA, 2025; HaCohen et al., 2026; 2024; NVIDIA, 2026; Hong et al., 2022) has pushed the boundaries of scalable video synthesis, enabling high-quality generation with strong temporal coherence. These architectures are built upon large-scale video diffusion transformers (DiTs), which process video tokens through a stack of self-attention layers (Vaswani et al., 2017). Despite these advances, generating physically plausible dynamics remains a prominent challenge. One reason could be that physical phenomena involve numerous simultaneously interacting elements, making it unclear whether DiTs alone can learn to simulate such complex dynamics.

Indeed, prior works address this challenge by incorporating external reasoning capabilities, such as (Vision-) Language Models (Xue et al., 2025; Yang et al., 2025c; Zhang et al., 2025b; Feng et al., 2026; Hsin-Ying et al., 2026; Huang et al., 2026), physical simulators (Foo et al., 2026), physics-informed adapters (Wang* et al., 2025; Yuan et al., 2026), or databases (Cheng et al., 2026). Some methods post-train video models via Direct Preference Optimization (DPO) and reinforcement learning (Cai et al., 2026; Zhao et al., 2026; Ji et al., 2025; Wu et al., 2025; Li et al., 2025; Zhang et al., 2026; Kang et al., 2024), using reward mechanisms derived from handcrafted physical knowledge or VLMs rather than video data alone.

Conversely, several works investigate learning physics purely through standard diffusion training objectives and video data (Gillman et al., 2026; 2025; Romero et al., 2026; Perini et al., 2026; Chao et al., 2026; Kang et al., 2024). These works construct synthetic datasets targeting specific physical phenomena (e.g., ball rolling and collision). Despite their intriguing results, they leave two critical questions unanswered. First, these works (as well as other existing benchmarks (Bansal et al., 2025; 2026; Guo et al., 2026; Gu et al., 2026; Motamed et al., 2025; Rädsch et al., 2026; Zhou et al., 2026)) typically address physical phenomena involving only a small number (e.g., 3) of interacting objects at a time. Real-world physical systems involve vastly more components acting concurrently and exhibit long dependencies, raising the question of whether DiTs can perform such *long-chain* reasoning. Second, these works often do not analyze whether models simply memorize and interpolate data patterns (Kang et al., 2024) or actually reason based on underlying physical laws. Uncovering these internal mechanisms is crucial for refining and justifying future model architectures and training strategies.

To the best of our knowledge, this paper is the first-of-its-kind attempt to address these questions. Using 2D gear systems as a testbed, we study video DiTs' ability to simulate simultaneously interacting objects. By analyzing the internal mechanisms of the trained DiTs, we demonstrate that video DiTs can reason over kinematic chain dependencies.

Algorithmic reasoning with vision generative models. Beyond photorealistic video generation, recent literature has applied image and video generative models to various problem-solving tasks (Wewer et al., 2025; Liu et al., 2024; Wiedemer et al., 2025b; Wang et al., 2026c;b; Tong et al., 2026; Suleymanzade et al., 2026; Drozdova et al., 2026; Xu et al., 2026; Luo et al., 2025; He et al., 2026; Yang et al., 2025b; Qi et al., 2026; Liu et al., 2025; Wang et al., 2026a; Geirhos et al., 2026; Zhu et al., 2026b; Guo et al., 2025; Newman et al., 2026; Liu et al., 2026; Chen et al., 2026; Kim et al., 2026; Jun et al., 2026; Yang et al., 2025a). A closely related task is maze solving, in which video DiTs are tasked with finding a continuous path to a goal while avoiding obstacles. Maze solving requires long-horizon reasoning to traverse the grid, and prior works (Newman et al., 2026; Cai et al., 2025) observe that the problem's difficulty scales with the length of the required path to the goal. Notably, previous approaches embed these path-traversal steps along the temporal dimension (i.e., Chain-of-Frames (CoF) (Wiedemer et al., 2025a)), with each traversal step supervised as a frame. To enable long sequential reasoning, several works (Chao et al., 2026; Newman et al., 2026) introduce an autoregressive formulation in which reasoning steps are decomposed across multiple generation passes. In contrast, we study a non-autoregressive reasoning setting in which long-range kinematic dependencies must be resolved simultaneously within a single generation process. Moreover, our models receive no supervision over the intermediate reasoning steps. Despite these constraints, we find that video DiTs can still learn effective reasoning mechanisms by leveraging their self-attention layers as a sequence of reasoning steps.

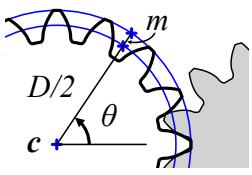
Graph-algorithmic reasoning with transformers. Since resolving kinematic chains can be cast as a graph traversal problem, our work is also related to theoretical studies on graph-algorithmic reasoning in generic transformer models. Prior theoretical work has shown that non-autoregressive transformers with shallow depth and polynomial width can efficiently execute sequential reasoning by learning parallelized *shortcut* algorithms. For instance, tasks such as simulating sequential state transitions in finite-state automata (FSAs) (Liu et al., 2023) or checking graph connectivity (Sanford et al., 2024) can be solved with logarithmic-depth transformers using a divide-and-conquer strategy. Conversely, other theoretical studies (Feng et al., 2023; Zhu et al., 2026a) investigate autoregressive formulations (i.e., *Chain-of-Thought* (CoT) reasoning), showing that unrolling intermediate reasoning steps as sequential tokens allows these problems to be solved with much smaller models. However, while theoretical work considers small, ad hoc models (e.g., 2-layer transformers) with specific input formats and handcrafted weights, our work empirically investigates whether large, off-the-shelf video diffusion transformers (Wan, 2025) operating on patchified video tokens can learn such efficient algorithms in practice.

3 Problem Formulation

We consider 2D involute gear systems as a testbed for evaluating the ability of video diffusion transformers to simulate systems of simultaneously interacting physical objects with kinematic chain dependencies. This section describes the design and motivation of this setup.

3.1 Preliminaries: 2D Involute Gear systems

We consider planar involute gear systems, i.e., collections of meshing gears constrained to a two-dimensional plane. Each *involute gear* is characterized by its *center* $\mathbf{c} = (c_x, c_y) \in \mathbb{R}^2$, *pitch diameter* $D \in \mathbb{R}_+$, and number of evenly spaced *teeth* $M \in \mathbb{N}$. The *module* $m = D/M$ determines the tooth size, and we assume that all gears share the same module m. The teeth extend beyond the pitch circle by the *addendum* $a = m$, giving rise to a *tip circle* with diameter $D + 2m$.



The gear system is represented by an undirected *kinematic graph* $G = (V, E)$, where the vertices $V = 1, \ldots, K$ index the gears, with gear i specified by its center $\mathbf{c}_i$ and pitch diameter D_i, and the edges E denote meshing pairs. We say that two gears *mesh* when their tip circles intersect. The

state of the system is described by the vector of rotation angles $\boldsymbol{\theta}(t) = (\theta_1(t), \ldots, \theta_K(t))$, where $\theta_i(t) \in \mathbb{R}$ is the rotation angle of gear i at time $t \in [0, 1]$, and $\theta_i(0) = 0$ for all i. Because meshing gears must maintain equal tangential velocities at their pitch circles, their angular velocities satisfy the *kinematic constraint* $D_i\dot{\theta}_i(t) = -D_j\dot{\theta}_j(t)$ for every pair of meshing gears $\{i, j\} \in E$. For simplicity, we assume G is a single connected tree.

3.2 Representing Gear systems as RGB videos

We reduce the problem of simulating gear systems to generating RGB videos $\mathbf{V} \in [-1, 1]^{F\times H\times W}$ of the motion of 2D involute gear systems. To isolate the challenges of kinematic reasoning from visual confounders (e.g., textures), we design a visually simple video representation $\mathbf{V}$ and develop an RGB encoding/decoding scheme that maps between a gear system $(G, \boldsymbol{\theta})$ and a video $\mathbf{V}$.

Encoding gear systems. We represent each gear $i \in V$ using its normalized object coordinates (NOCs) (Wang et al., 2019). Specifically, for the first frame, each surface point is represented as $\boldsymbol{r} = [r_x, r_y]^\top \in [-1, 1]^2$, defined relative to the gear center $\boldsymbol{c}_i$ and tip radius $R_i = (D_i + 2a)/2$. These coordinates are then transformed across subsequent frames according to the underlying gear motion. As illustrated in Fig. 1, this representation enables dense tracking of the gear surface throughout the video. Further details are provided in Sec. B.

Decoding videos. We develop an algorithm to recover the gear system $(G, \boldsymbol{\theta})$ from the generated video $\mathbf{V}$ by estimating the gear centers $\boldsymbol{c}_i$, tip radii R_i, and per-frame rotation angles $\theta_i(t)$. Owing to the normalized form of our video representation, these quantities can be recovered by solving a least-squares problem over a sliding spatial window (see Sec. C for details).

3.3 Problem Setup: Simulating Gear Systems

Building on the gear representation developed above, we task video diffusion transformers (DiTs) with simulating the motion of a gear system. Specifically, as illustrated in Fig. 1, the model is provided with the initial spatial layout of a gear system $G = (V, E)$ as the first frame. A single gear $g \in V$ is designated as the **driving gear**, and its full rotational trajectory $\theta_g(t)$ is provided as a conditioning video. The objective is to synthesize a video $\hat{\mathbf{V}}$ of all gears in the system while satisfying the underlying kinematic constraints, which are then parsed to obtain the estimated motion $\hat{\boldsymbol{\theta}}$. We fine-tune the Wan2.1 (1.3B) text-to-video model (Wan, 2025). Implementation details are provided in Sec. D.

Why study gear simulation? Gear systems are governed by a simple local rule: meshed gears rotate in opposite directions, with absolute angular velocities inversely proportional to their diameters. Despite this simple rule, simulating a gear system provides a challenging testbed for kinematic chain reasoning. In particular, determining the rotational direction of each gear requires computing its rotational *parity*, which is determined by the number of meshing interactions along the kinematic path from the driving gear. Thus, simulating a gear system requires propagating kinematic information across potentially long chains of gears. Notably, once the rotational parity is determined, the angular velocity magnitude follows directly from the gear diameters, via $|\dot{\theta}_i(t)| = |\dot{\theta}_g(t)|D_g/D_i$, independent of the kinematic graph topologies. As illustrated in Fig. 2, off-the-shelf video generation models struggle to synthesize kinematically plausible animations of gear systems, even for the simple case of two meshing gears, where the gears rotate in conflicting directions.

4 Method

In this section, we introduce the key ingredients at the basis of our study: data generation and evaluation (Sec. 4.1), and feature probing (Sec. 4.2).

4.1 Dataset generation and evaluation

We synthesize gear systems starting from a single gear and iteratively selecting a parent gear and attaching to it a new gear at a random angle, avoiding collisions. To avoid gear size being a confounding factor in our analysis, we apply resizing augmentation during training and constrain the

minimum gear radius at inference time (see Sec. A.3). Since selecting parent gears uniformly at random tends to result in shallow kinematic height (see Fig. 7a), we instead sample unfiromly the kinematic height and construct the gear system accordingly. Details are provided in Sec. A.2.

Our primary focus is assessing the model's ability to determine the rotational parity of each gear relative to the driving gear—the most challenging aspect of our task, as discussed in Sec. 3.3. We quantify this capability using the following two metrics.

Relative Motion Discrepancy ($\mathcal{E}_{\text{rmd}}$). We quantify the motion discrepancy between the ground-truth gear system G and the one recovered from the generated video $\hat{G}$ by computing the normalized squared error between the generated and ground-truth angular velocities:

$$\mathcal{E}_{\text{rmd}} = \frac{1}{K}\sum_{i=1}^{K} \frac{\sum_{t=2}^{T}\left(D_i\Delta\theta_{i,t} - \hat{D}_{\sigma^*(i)}\Delta\hat{\theta}_{\sigma^*(i),t}\right)^2}{\sum_{t=2}^{T}(D_i\Delta\theta_{i,t})^2 + \epsilon},$$

where K is the number of gears and $\sigma^*(i)$ maps the i-th ground-truth gear to its counterpart in $\hat{G}$. The term $\Delta\theta_{i,t} = \theta_{i,t} - \theta_{i,t-1}$ represents the discrete angular velocity and $\epsilon > 0$ is a small constant for numerical stability. An $\mathcal{E}_{\text{rmd}}$ of 0 indicates a perfect motion match. Conversely, if the generated gear matches the reference speed but rotates in the opposite direction, $\mathcal{E}_{\text{rmd}} = 4$ (assuming $\epsilon = 0$).

Simulation Success Rate (SSR). A generated gear system is considered successful if and only if every gear exhibits the correct rotational motion. Specifically, we compute the Relative Motion Disparity ($\mathcal{E}_{\text{rmd}}$) for each gear individually, without averaging across gears. A test sample is considered successful if the error of every gear is below a tolerance threshold $\tau = 0.25$. We report SSR as the fraction of successful test samples.

4.2 Feature probing

This section introduces PCA-based analysis and MLP probing techniques that allow us to analyze the internal mechanisms of video DiTs for determining the rotational parity of each gear.

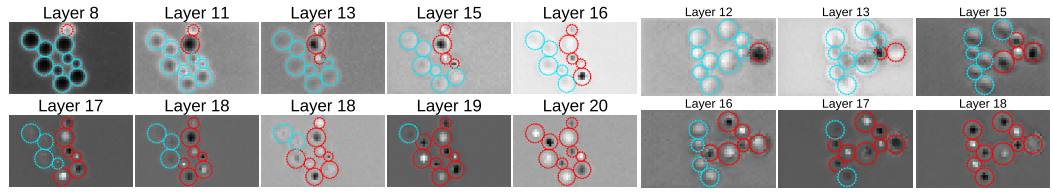
PCA analysis allows us to visualize how video DiTs represent kinematic information. Given a spatial-temporal feature map $F \in \mathbb{R}^{C\times T\times H\times W}$ extracted from attention heads, we flatten it into $N = T \times H \times W$ tokens, apply PCA, and visualize the leading principal components. As shown in Fig. 3, we identify principal axes that cleanly separate gears with different rotational parities.

Building on the PCA results, we train lightweight MLPs that predict whether a pair of feature vectors corresponding to two distinct gears has the same rotational parity or not. The probing network is symmetric: each feature vector is first processed independently by MLP layers with shared weights, the resulting features are combined via element-wise multiplication, and the result is passed through another set of MLP layers. Notably, parity information is not uniformly encoded across time. Therefore, rather than pooling the feature vectors across time into a single vector ($x \in \mathbb{R}^C$), we concatenate them along the temporal dimension for probing ($x \in \mathbb{R}^{(T\cdot C)}$). We discuss the implications of this temporal variation further in Sec. E.3. Architecture details are provided in Sec. D and Sec. E.1.

5 Experiments

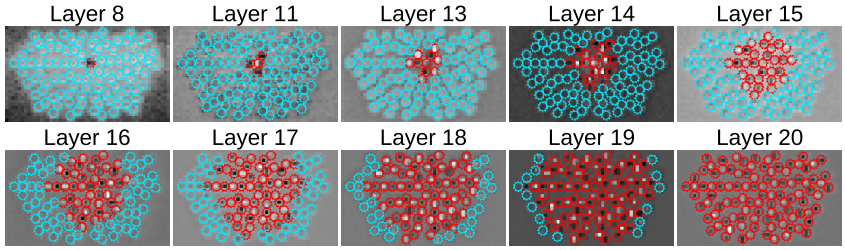
Using the machinery introduced above, we investigate the ability of video diffusion transformers (DiTs) to perform long-chain kinematic reasoning, with a particular focus on determining the rotational parity of each gear. We find that DiTs can indeed perform such kinematic chain reasoning very well. Crucially, we identify two distinct reasoning mechanisms that emerge depending on the *kinematic height* h (i.e., the maximum depth from the driving gear) observed during training.

When the training data contains only small kinematic heights, the model learns a parallel BFS strategy, in which successive groups of transformer layers propagate information outward from the driving gear, one depth at a time. In contrast, when the model is exposed to larger kinematic heights, for which a parallel BFS strategy cannot be implemented due to the limited number of layers, it instead learns a divide-and-conquer-like strategy, in which local parities are first determined and subsequently merged. The following two sections, Sec. 5.1 and Sec. 5.2, demonstrate these two reasoning mechanisms, respectively.

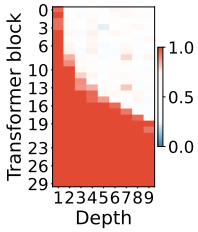


(a) Trained on: $N \leq 10, h \leq 10$, General Kinematic Tree

(b) Trained on: $N, h \leq 10$, Linear Only



(c) Trained on: $N, h \leq 10$ only, General Kinematic Tree (Zoom-in recommended)



(d) MLP probing

Figure 3: **Emergence of parallel-BFS-like parity determination across layers. (a) PCA analysis on linear kinematic chains.** Models are trained and tested on up to 10 gears. PCA visualizations are overlaid with the contours of the gears, where we color the gears in red according to the ground-truth parallel BFS mechanism. **(b) PCA analysis on unseen kinematic topologies.** Models are trained solely on linear graphs, but tested on non-linear kinematic topologies. **(c) PCA analysis on O.O.D. gear counts.** Models are trained on up to 10 gears, but tested on systems with 100 gears and in-distribution height $h \leq 10$. **(d) MLP probing accuracy.** Probing accuracy across transformer layers for models trained on up to 10 gears and evaluated on a linear kinematic chain of $N = h = 10$. (See Fig. 16 for models trained on up to 5 gears.) Together, these results show parallel-BFS-like, depth-by-depth determination of rotational parity across transformer layers.

5.1 Emergence of Parallel BFS-Like Reasoning

We begin by demonstrating the emergence of parallel BFS reasoning by training a video DiT on gear mechanisms with gear counts N and kinematic heights h up to 10, which require only short-chain reasoning. Notably, this maximum height is smaller than the number of transformer layers (30).

The PCA analysis (Fig. 3) reveals an interesting pattern: there is a principal axis that separates features associated with gears of different rotational parity. Remarkably, this parity signal first emerges around the driving gear, and then progressively propagates to neighboring gears across successive transformer layers, resembling a parallel BFS-like progression. To quantitatively verify this process, we perform MLP probing to measure the amount of information each transformer layer stores about the rotational parity of individual gears relative to the driving gear. As shown in Fig. 3d, the probing accuracy increases almost linearly with network depth, providing evidence that gear parity is determined incrementally across layers.

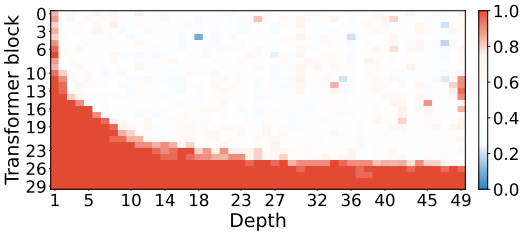
We note several other properties. First, the learned algorithm generalizes poorly to unseen kinematic heights. As shown in Tab. 1, the model achieves a success rate of almost 0 on out-of-distribution cases when the kinematic height exceeds the training range. Feature probing (Fig. 4) also reveals that the model successfully propagates parity information across transformer layers up to the height observed during training, but that this iterative propagation ceases beyond that height. In contrast, the model can generalize to unseen gear counts and branching factors as long as the kinematic height remains within the training distribution. As shown in Tab. 1 and Fig. 3c, the model performs well on mechanisms containing 100 gears, despite be-

Table 2: **Generalization to unseen tree topologies.** Simulation success rate (SSR) and Relative Motion Disparity ($\mathcal{E}_{\text{rmd}}$) for models trained exclusively on linear kinematic chains ($N, h \leq 5, 10$) and evaluated on linear chains (I.D.) versus general kinematic trees (O.O.D.).

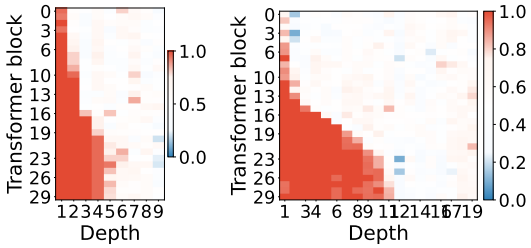
	Linear Chain (I.D.)		General Tree (O.O.D.)	
Model	**SSR ↑**	$\mathcal{E}_{\text{rmd}}$	**SSR ↑**	$\mathcal{E}_{\text{rmd}}$
$N_{\text{train}} \leq 5$	100%	0.058	100%	0.058
$N_{\text{train}} \leq 10$	97%	0.046	99%	0.040

Table 1: **Simulation success rates (%) across varying gear counts and kinematic heights.** We evaluate models trained on up to 10 gears. The in-distribution settings are enclosed in **red**. Performance remains high for out-of-distribution gear counts (N) as long as the kinematic height (h) remains within the training distribution. **See Tab. 6 for models trained on up to 5 gears.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	100%	—	—	—	—	—	—	—	—	—
20	100%	1%	—	—	—	—	—	—	—	—
30	97%	1%	0%	—	—	—	—	—	—	—
40	100%	0%	0%	0%	—	—	—	—	—	—
50	94%	0%	0%	0%	0%	—	—	—	—	—
60	90%	0%	0%	0%	0%	0%	—	—	—	—
70	100%	0%	0%	0%	0%	0%	0%	—	—	—
80	99%	0%	0%	0%	0%	0%	0%	0%	—	—
90	89%	0%	0%	0%	0%	0%	0%	0%	0%	—
100	82%	0%	0%	0%	0%	0%	0%	0%	0%	0%

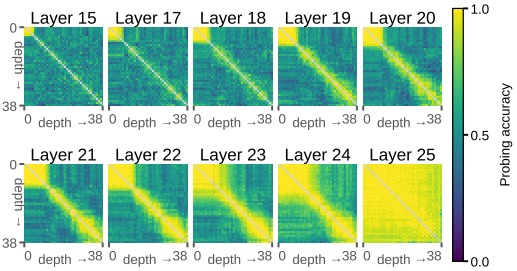


(a) Parity prediction accuracy across transformer layers and kinematic depths.



(a) Train: $H \leq 5$ Eval: $H = 10$ (b) Train: $H \leq 10$ Eval: $H = 20$

Figure 4: **Feature probing on out-of-distribution kinematic heights.** Models trained on mechanisms with heights up to $5/10$ and evaluated on a $10/20$-gear linear chain. Parity propagation extends only up to the in-distribution height, beyond which it stalls.



(b) Pairwise parity prediction accuracy across transformer layers. **See the webpage and Fig. 28.**

Figure 5: **Emergence of a divide-and-conquer-like parity determination strategy. (a)** MLP probing accuracy for parity relative to the driving gear across transformer layers and kinematic depths. **(b)** Pairwise parity prediction accuracy across the linear gear chain (full results in Sec. I). Together, these results suggest that the model first resolves parity within local neighborhoods and progressively combines these local solutions into a globally consistent representation.

ing trained only on mechanisms with up to 10 gears, provided that the kinematic height remains in-distribution.

Furthermore, when we train the model exclusively on gear mechanisms forming linear kinematic chains, in which each interior gear has exactly two neighboring gears (see Fig. 8 for examples), the trained model still performs well on arbitrary tree topologies during inference (Tab. 2). Fig. 3b shows the PCA analysis for this setting, where the transformer layers successfully propagate parity information to previously unseen numbers of neighboring gears in parallel.

These behaviors can be naturally explained. The transformer learns to activate only the number of layers necessary to perform enough BFS steps to cover the kinematic height encountered during training, which limits generalization to O.O.D. kinematic heights. However, the parallel nature of the attention mechanism allows the model to generalize across different branching factors, enabling extrapolation to previously unseen tree topologies as long as their kinematic heights remain within the training distribution.

5.2 Emergence of Divide-and-Conquer-Like Reasoning

We next demonstrate the emergence of divide-and-conquer-like reasoning by training models on gear mechanisms with numbers of gears and kinematic heights up to 50. Notably, our video DiT contains only 30 transformer layers. Therefore, a straightforward parallel BFS strategy is infeasible.

Our PCA analysis in Fig. 6 reveals an intriguing departure from the short-chain setting. In the early layers, we continue to observe the parallel BFS pattern. However, in mid-

late layers, parity signals begin to emerge simultaneously in gears that have not yet been reached by the BFS process. Importantly, these newly emerging signals are initially consistent only within their respective neighborhoods, but subsequently become globally consistent. MLP probing (Fig. 5a) provides further evidence for this behavior. In the shallow layers, probing accuracy increases approximately linearly with kinematic depth. In later layers, however, the accuracy increases rapidly. To further investigate this mechanism, Fig. 5b visualizes the per-layer pairwise parity prediction accuracy across the linear gear chain. We observe the formation of clusters in which parity predictions are highly accurate locally but remain globally inconsistent. In later layers, these local clusters progressively merge, ultimately yielding a globally consistent parity representation.

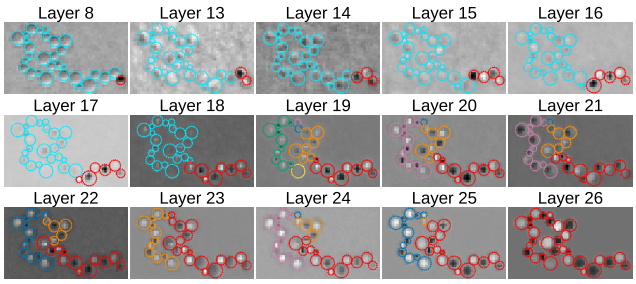


Figure 6: **Emergence of a divide-and-conquer strategy.** The model is trained on gear chains of size $N, h \leq 30$ and evaluated on $N = h = 30$. Gears are grouped and color-coded by manual inspection according to their local parity consistency in later layers.

Taken together, these observations suggest that the model first establishes parity relationships within local neighborhoods and subsequently combines these locally consistent regions, effectively implementing a *divide-and-conquer-like* strategy. This mechanism allows the model to perform long-chain reasoning within a limited number of transformer layers.

However, the learned divide-and-conquer strategy does not exhibit the same degree of generalizability as the parallel BFS strategy. As shown in Tab. 3, we observe weaker generalization to unseen gear counts, even when the kinematic height remains within the training distribution. The main exception is when the kinematic height is up to around 10, where the model continues to generalize. This is because the model can still rely on parallel BFS reasoning. Another notable difference is that the model starts exhibiting non-zero success rates on unseen kinematic heights, which was not the case in experiments with shorter kinematic chains. These findings suggest that the learned divide-and-conquer-like strategies are entangled with statistics of various topological factors in the training data distribution. This contrasts with short-chain experiments, where performance depends almost solely on kinematic height. Consequently, while the model exhibits extrapolation in certain cases, the strategy is still not universally generalizable.

Table 3: **Simulation success rates (%) for model trained up to 50 gears.** In-distribution configurations ($N, h \leq 50$) are enclosed in **red**.

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	100%	—	—	—	—	—	—	—	—	—
20	100%	100%	—	—	—	—	—	—	—	—
30	99%	100%	100%	—	—	—	—	—	—	—
40	99%	100%	100%	100%	—	—	—	—	—	—
50	99%	99%	97%	98%	95%	—	—	—	—	—
60	99%	95%	97%	96%	95%	85%	—	—	—	—
70	100%	99%	89%	67%	56%	54%	43%	—	—	—
80	100%	97%	84%	62%	57%	42%	35%	31%	—	—
90	100%	90%	74%	63%	46%	38%	29%	25%	19%	—
100	99%	92%	73%	47%	35%	20%	23%	13%	15%	5%

6 Discussion and Conclusion

In this work, we investigate the ability of video diffusion transformers (DiTs) to perform kinematic chain reasoning through a novel problem setup based on 2D involute gear systems. Our analysis shows that video DiTs learn specific reasoning mechanisms that allow them to correctly determine rotational parity across kinematic chains, even when their architectural depth appears insufficient to support straightforward linear reasoning. However, these learned reasoning mechanisms exhibit limited extrapolation beyond the complexity encountered during training.

Although conducted in a simplified setting, our study demonstrates the ability of video DiTs to learn algorithmic reasoning. However, these models tend to adopt the *easiest* reasoning strategy that suffices for the training distribution, suggesting that simply scaling up the training data does not necessarily lead to more generalizable reasoning. Therefore, developing complexity-aware data collection methods and designing architectural inductive biases that encourage scalable reasoning strategies represent promising directions for future work.

Finally, our setting focuses on a class of causal reasoning problems whose solutions can be efficiently composed through local operations. Investigating less decomposable scenarios remains an interesting open question for understanding the limits of algorithmic reasoning in video DiTs.

ETHICS STATEMENT

This work does not involve human subjects, personal or sensitive data, or human-generated annotations. Our dataset is procedurally generated via a hand-crafted program, avoiding privacy and consent concerns associated with real-world data collection. We are not aware of any specific ethical risks or harmful applications directly arising from our work. However, we anticipate that future advances in video diffusion transformers may enable increasingly powerful physical reasoning capabilities, which could be misused in unforeseen ways. We advocate against the misuse of generative models for malicious or deceptive purposes and encourage their safe and responsible development and use.

REPRODUCIBILITY STATEMENT

We provide the implementation details needed to reproduce our results throughout the paper and supplementary materials. Our work builds on the open-source video diffusion transformer Wan (2025), and our dataset is generated procedurally rather than relying on private data. The code will be publicly released upon acceptance, including the dataset generation and probing procedures.

LLM USAGE STATEMENT

ChatGPT and Gemini were used to assist with manuscript writing. CodeX was used to assist with coding, particularly for debugging and generating visualization results. LLMs were not involved in the ideation or development of the research. The authors have carefully reviewed and verified all LLM-assisted content and take full responsibility for the scientific integrity of this work.

REFERENCES

Artiprocher and ModelScope AIGC Team. Diffsynth-studio: An open-source diffusion engine for image and video generation. https://github.com/modelscope/DiffSynth-Studio, 2024. Version 0.1.0 and later.

Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. VideoPhy: Evaluating physical commonsense for video generation. *Proc. ICLR*, 2025.

Hritik Bansal, Clark Peng, Yonatan Bitton, Roman Goldenberg, Aditya Grover, and Kai-Wei Chang. VideoPhy-2: A challenging action-centric physical commonsense evaluation in video generation. In *Proc. ICLR*, 2026.

Yuanhao Cai, Kunpeng Li, Menglin Jia, Jialiang Wang, Junzhe Sun, Feng Liang, Weifeng Chen, Felix Juefei-Xu, Chu Wang, Ali Thabet, Xiaoliang Dai, Xuan Ju, Alan Yuille, and Ji Hou. Phygdpo: Physics-aware groupwise direct preference optimization for physically consistent text-to-video generation. In *ECCV*, 2026.

Zefan Cai, Haoyi Qiu, Tianyi Ma, Haozhe Zhao, Gengze Zhou, Kung-Hsiang Huang, Parisa Kordjamshidi, Minjia Zhang, Wen Xiao, Jiuxiang Gu, Nanyun Peng, and Junjie Hu. Mmgr: Multimodal generative reasoning, 2025. URL https://arxiv.org/abs/2512.14691.

Jorge Diaz Chao, Konpat Preechakul, Yuxi Liu, and Yutong Bai. The seriality gap in video diffusion models. *arXiv preprint arXiv:2607.13031*, 2026.

Xinyan Chen, Ziyu Guo, Renrui Zhang, Dongzhi Jiang, and Hongsheng Li. Opencof: Learning to reason through video generation. *arXiv preprint arXiv:2607.08763*, 2026.

Yongfan Chen, Xiuwen Zhu, and Tianyu Li. A physical coherence benchmark for evaluating video generation models via optical flow-guided frame prediction. *arXiv*, 2025.

Kexu Cheng, Zicheng Liu, Mingju Gao, Chunhe Song, and Hao Tang. Physrag: Enhancing physics-awareness in video generation via retrieval-augmented generation, 2026. URL https://arxiv.org/abs/2606.26916.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1070. URL https://aclanthology.org/D17-1070/.

Mariia Drozdova, Aidan Sirbu, Pietro Miotti, Robert Obryk, Mayalen Etcheverry, Eyvind Niklasson, and Blake Richards. Diffusion as a training curriculum for timestep-free iterative reasoning, 2026. URL https://arxiv.org/abs/2609.01449.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=qHrADgAdYu.

Yuxiang Feng, Juncheng Wang, Chao Xu, Yijie Qian, Huihan Wang, Wenlong Hou, Yang Liu, Baigui Sun, Yong Liu, and Shujun Wang. Newton: Agentic planning for physically grounded video generation. *arXiv preprint arXiv:2605.18396*, 2026. URL https://arxiv.org/abs/2605.18396.

Lin Geng Foo, Mark He Huang, Alexandros Lattas, Stylianos Moschoglou, Thabo Beeler, and Christian Theobalt. Physical simulator in-the-loop video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4301–4311, June 2026.

Robert Geirhos, Yuxuan Li, Thaddäus Wiedemer, Neha Kalibhat, Zi Wang, Mani Malek, Øyvind Tafjord, Kevin Swersky, Been Kim, and Priyank Jaini. Visual prompt engineering for video models. *arXiv preprint arXiv:2607.25537*, 2026.

Genmo Team. Mochi 1. https://github.com/genmoai/models, 2024.

Nate Gillman, Charles Herrmann, Michael Freeman, Daksh Aggarwal, Evan Luo, Deqing Sun, and Chen Sun. Force prompting: Video generation models can learn and generalize physics-based control signals. 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, Main Conference, pp. 103174–103201. Curran Associates, Inc., 2025. doi: 10.52202/085713-3448. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/953dcf20cbd275465066ad63ea111f13-Paper-Conference.pdf.

Nate Gillman, Yinghua Zhou, Zitian Tang, Evan Luo, Arjan Chakravarthy, Daksh Aggarwal, Michael Freeman, and Chen Sun. Goal force: Teaching video models to accomplish physics-conditioned goals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20077–20087, June 2026.

Google DeepMind. Veo 3.1. https://deepmind.google/models/veo/, 2026. Accessed 2026-05-13.

Jing Gu, Xian Liu, Yu Zeng, Ashwin Nagarajan, Fangrui Zhu, Daniel Hong, Yue Fan, Qianqi Yan, Kaiwen Zhou, Ming-Yu Liu, and Xin Eric Wang. PhyWorldBench: A comprehensive evaluation of physical realism in text-to-video models. In *Proc. ICLR*, 2026.

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. *arXiv*, 2026.

Ziyu Guo, Xinyan Chen, Renrui Zhang, Ruichuan An, Yu Qi, Dongzhi Jiang, Xiangtai Li, Manyuan Zhang, Hongsheng Li, and Pheng-Ann Heng. Are video models ready as zero-shot reasoners? an empirical study with the mme-cof benchmark. *arXiv preprint arXiv:2510.26802*, 2025.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion. 2024. URL https://arxiv.org/abs/2501.00103.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. Ltx-2: Efficient joint audio-visual foundation model. 2026. URL https://arxiv.org/abs/2601.03233.

Xuan He, Cong Wei, Yuhao Cheng, Linrui Ma, Yuxuan Zhang, Zuojun Li, Yuhao Wen, Jize Jiang, Zeyi Liu, Yuren Hao, Songcheng Cai, Keming Wu, Penghui Du, Kai Zou, Rui Yang, Chenkai Sun, Ke Yang, Ping Nie, Kelsey R Allen, Chenglong Wang, Michel Galley, Jianfeng Gao, and ChengXiang Zhai. Vgi-bench: Probing visual intelligence in video generation models, 2026. URL https://arxiv.org/abs/2608.19583.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

Lee Hsin-Ying, Hanwen Jiang, Yiqun Mei, Jing Shi, Ming-Hsuan Yang, and Zhixin Shu. Motimotion: Motion-controlled video generation with visual reasoning. In *Forty-third International Conference on Machine Learning*, 2026.

Ziqi Huang, Ning Yu, Gordon Chen, Haonan Qiu, Paul Debevec, and Ziwei Liu. VChain: Chain-of-visual-thought for reasoning in video generation. In *Annual Meeting of the Association for Computational Linguistics (ACL Findings), 2026*, 2026.

Sangwon Jang, Taekyung Ki, Jaehyeong Jo, Saining Xie, Jaehong Yoon, and Sung Ju Hwang. Self-refining video sampling. *arXiv preprint arXiv:2601.18577*, 2026.

Sihui Ji, Xi Chen, Xin Tao, Pengfei Wan, and Hengshuang Zhao. PhysMaster: Mastering Physical Representation for Video Generation via Reinforcement Learning. *arXiv preprint arXiv:2510.13809*, 2025.

Sejoon Jun, Zheng Ding, Huangyuan Su, Weirui Ye, and Yilun Du. Temporal backtracking search for test-time generative video reasoning, 2026. URL https://arxiv.org/abs/2606.13861.

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. *arXiv preprint arXiv:2411.02385*, 2024.

Joowon Kim, Seungho Shin, Joonhyung Park, and Eunho Yang. Collabvr: Collaborative video reasoning with vision-language and video generation models. *arXiv preprint*, 2026.

Weijie Kong et al. Hunyuanvideo: A systematic framework for large video generative models, 2025. URL https://arxiv.org/abs/2412.03603.

Chenyu Li, Oscar Michel, Xichen Pan, Sainan Liu, Mike Roberts, and Saining Xie. PISA experiments: Exploring physics post-training for video diffusion models by watching stuff drop. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=RFCp1QzzHQ.

Wang Lin, Liyu Jia, Wentao Hu, Kaihang Pan, Zhongqi Yue, Wei Zhao, Jingyuan Chen, Fei Wu, and Hanwang Zhang. Reasoning physical video generation with diffusion timestep tokens via reinforcement learning. *arXiv*, 2025.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

Jinyang Liu, Wondmgezahu Teshome, Sandesh Ghimire, Mario Sznaier, and Octavia Camps. Solving masked jigsaw puzzles with diffusion vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23009–23018, 2024.

Ming Liu, Yunbei Zhang, Shilong Liu, Liwen Wang, and Wensheng Zhang. Wan-r1: Verifiable-reinforcement learning for video reasoning, 2026. URL https://arxiv.org/abs/2603.27866.

Xinxin Liu, Zhaopan Xu, Kai Wang, Yong Jae Lee, and Yuzhang Shang. Can world simulators reason? gen-vire: A generative visual reasoning benchmark. *arXiv preprint arXiv:2511.13853*, 2025.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Luma AI. Ray3: Ai video generation with hdr, motion & visual reasoning. https://lumalabs.ai/ray, 2025. Accessed: 2025-11-06.

Luma AI. Ray 3.14. https://lumalabs.ai/ray, 2026. Accessed 2026-05-13.

Yang Luo, Xuanlei Zhao, Baijiong Lin, Lingting Zhu, Liyao Tang, Yuqi Liu, Ying-Cong Chen, Shengju Qian, Xin Wang, and Yang You. V-reasonbench: Toward unified reasoning benchmark suite for video generation models, 2025. URL https://arxiv.org/abs/2511.16668.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models learn physical principles from watching videos? *arXiv*, 2501.09038, 2025.

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.

NVIDIA. World simulation with video foundation models for physical ai, 2025. URL https://arxiv.org/abs/2511.00062.

NVIDIA. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026. URL https://research.nvidia.com/labs/cosmos-lab/cosmos3/technical-report.pdf.

Janne Perini, Rafael Bischof, Moab Arar, Ayça Duran, Michael A. Kraus, Siddhartha Mishra, and Bernd Bickel. Pretrained video models as differentiable physics simulators for urban wind flows. *arXiv preprint arXiv:2603.21210*, 2026.

Yu Qi, Xinyi Xu, Ziyu Guo, Siyuan Ma, Renrui Zhang, Xinyan Chen, Ruichuan An, Ruofan Xing, Jiayi Zhang, Haojie Huang, et al. Mme-cof-pro: Evaluating reasoning coherence in video generative models with text and visual hints. *arXiv preprint arXiv:2603.20194*, 2026.

David Romero, Ariana Bermudez, Viacheslav Iablochnikov, Hao Li, Fabio Pizzati, and Ivan Laptev. Learning to generate rigid body interactions with video diffusion models, 2026. URL https://arxiv.org/abs/2510.02284.

Tim Rädsch, Yuki M. Asano, Hilde Kuehne, Stefan Bauer, Priyank Jaini, Robert Geirhos, and Carsten T. Lüth. Physics-iq verified. *arXiv preprint arXiv:2606.18943*, 2026.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=AfzbDw6DSp.

Abhishek Sharma, Adams Yu, Ali Razavi, Andeep Toor, Andrew Pierson, Ankush Gupta, Austin Waters, Daniel Tanis, Dumitru Erhan, Eric Lau, Eleni Shaw, Gabe Barth-Maron, Greg Shaw, Han Zhang, Henna Nandwani, Hernan Moraldo, Hyunjik Kim, Irina Blok, Jakob Bauer, Jeff Donahue, Junyoung Chung, Kory Mathewson, Kurtis David, Lasse Espeholt, Marc van Zee, Matt McGill, Medhini Narasimhan, Miaosen Wang, Mikołaj Bińkowski, Mohammad Babaeizadeh, Mohammad Taghi Saffar, Nick Pezzotti, Pieter-Jan Kindermans, Poorva Rane, Rachel Hornung, Robert Riachi, Ruben Villegas, Rui Qian, Sander Dieleman, Serena Zhang, Serkan Cabi, Shixin Luo, Shlomi Fruchter, Signe Nørly, Srivatsan Srinivasan, Tobias Pfaff, Tom Hume, Vikas Verma, Weizhe Hua, William Zhu, Xinchen Yan, Xinyu Wang, Yelin Kim, Yuqing Du, and Yutian Chen. Veo, 2024. URL https://deepmind.google/technologies/veo/.

Ayhan Suleymanzade, Chanhyuk Lee, Floor Eijkelboom, Nicholas M. Boffi, İsmail İlkan Ceylan, and Jinwoo Kim. Thinking with looped flows, 2026. URL https://arxiv.org/abs/2609.11801.

Haoran Sun, Qingying Gao, Haiyun Lyu, Dezhi Luo, Yijiang Li, and Hokin Deng. Probing mechanical reasoning in large vision language models. In *Proc. ICLR Workshop*, 2025.

Jingqi Tong, Yurong Mou, Hangcheng Li, Mingzhe Li, Yongzhuo Yang, Ming Zhang, Qiguang Chen, Tianyi Liang, Xiaomeng Hu, Yining Zheng, Xinchi Chen, Jun Zhao, Xuanjing Huang, and Xipeng Qiu. Thinking with video: Video generation as a promising multimodal reasoning paradigm. In *Proc. CVPR*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NeurIPS*, 2017.

Team Wan. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Chen Wang*, Chuhao Chen*, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. In *NeurIPS*, 2025.

He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proc. CVPR*, 2019.

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, Yu Fengyuan, Weihang Xiao, Yizheng Jiao, Jianheng Hou, Danyang Zhang, Pengcheng Xu, Boyang ZHONG, Zehong Zhao, Gaoyun Fang, John Kitaoka, Xu Yile, 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, Dahua Lin, Ziwei Liu, Vikash Kumar, Yijiang Li, Lei Yang, Zhongang Cai, and Hokin Deng. A very big video reasoning suite. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=AwC77yHpP6.

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, 2026b. 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. Demystifing video reasoning. *arXiv preprint arXiv:2603.16870*, 2026c.

Christopher Wewer, Bart Pogodzinski, Bernt Schiele, and Jan Eric Lenssen. Spatial reasoning with denoising models. *arXiv preprint arXiv:2502.21075*, 2025.

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. *arXiv preprint arXiv:2509.20328*, 2025a.

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. *arXiv*, 2025b.

Ziyi Wu, Anil Kag, Ivan Skorokhodov, Willi Menapace, Ashkan Mirzaei, Igor Gilitschenski, Sergey Tulyakov, and Aliaksandr Siarohin. DenseDPO: Fine-grained temporal preference optimization for video diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=YFa7eULIeN.

Junxiang Xu, Ruisi Wang, Fanyi Pu, Maijunxian Wang, Ran Ji, Tongxi Zhou, Chenyang Gu, Jing Zuo, Hongcan Xiao, Yimeng Geng, Wanqi Yin, Chen Wei, Oscar Qian, Zhengan Yan, Ziqi Huang, Haiwen Diao, Liang Pan, Bo Li, Xiangyu Fan, Dezhi Luo, Fengyuan Yu, Zehong Zhao, Qingying Gao, Tinghui Zhu, Yilan Zhang, Jingqi Tong, Pinyuan Feng, Zhengze Jiang, Letian Wang, Ziyu Guo, Renrui Zhang, Jieneng Chen, Sonia Joseph, Constantin Venhoff, Saman Motamed, Mengyue Yang, Chandra Sripada, Alan Yuille, Philip Torr, Lvmin Zhang, Vikash Kumar, Daniel Khashabi, Nikolaus Kriegeskorte, Raphaël Millière, Vincent C. Müller, Anyi Rao, Quan Wang, Ziwei Liu, Dahua Lin, Lei Yang, Hokin Deng, and Zhongang Cai. Vbvr-pro: A scalable and verifiable suite for native visual reasoning. *arXiv preprint arXiv:2608.26105*, 2026. URL https://arxiv.org/abs/2608.26105.

Qiyao Xue, Xiangyu Yin, Boyuan Yang, and Wei Gao. Phyt2v: Llm-guided iterative self-refinement for physics-grounded text-to-video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18826–18836, 2025. doi: 10.1109/CVPR52734.2025.01754.

Cheng Yang, Haiyuan Wan, Yiran Peng, Xin Cheng, Zhaoyang Yu, Jiayi Zhang, Junchi Yu, Xinlei Yu, Xiawu Zheng, Dongzhan Zhou, and Chenglin Wu. Reasoning via video: The first evaluation of video models' reasoning abilities through maze-solving tasks, 2025a. URL https://arxiv.org/abs/2511.15065.

Cheng Yang, Haiyuan Wan, Yiran Peng, Xin Cheng, Zhaoyang Yu, Jiayi Zhang, Junchi Yu, Xinlei Yu, Xiawu Zheng, Dongzhan Zhou, and Chenglin Wu. Reasoning via video: The first evaluation of video models' reasoning abilities through maze-solving tasks. *arXiv preprint arXiv:2511.15065*, 2025b.

Xindi Yang, Baolu Li, Yiming Zhang, Zhenfei Yin, Lei Bai, Liqian Ma, Zhiyong Wang, Jianfei Cai, Tien-Tsin Wong, Huchuan Lu, and Xu Jia. Vlipp: Towards physically plausible video generation with vision and language informed physical prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12360–12370, October 2025c.

Yu Yuan, Xijun Wang, Tharindu Wickremasinghe, Zeeshan Nadir, Bole Ma, and Stanley H. Chan. Newtongen: Physics-consistent and controllable text-to-video generation via neural newtonian dynamics. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rJ6N6sunaU.

Chenyu Zhang, Daniil Cherniavskii, Antonios Tragoudaras, Antonios Vozikis, Thijmen Nijdam, Derck W. E. Prinzhorn, Mark Bodracska, Nicu Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Morpheus: Benchmarking physical reasoning of video generative models with real physical experiments. *arXiv*, 2025a.

Ke Zhang, Cihan Xiao, Yiqun Mei, Jiacong Xu, and Vishal M. Patel. Think before you diffuse: Llms-guided physics-aware video generation, 2025b. URL https://arxiv.org/abs/2505.21653.

Qiyuan Zhang, Biao Gong, Shuai Tan, Zheng Zhang, Yujun Shen, Xing Zhu, Yuyuan Li, Kelu Yao, Chunhua Shen, and Changqing Zou. Physrvg: Physics-aware unified reinforcement learning for video generative models. *ECCV 2026*, 2026.

Pu Zhao, Juyi Lin, Timothy Rupprecht, Arash Akbari, Chence Yang, Rahul Chowdhury, Elaheh Motamedi, Arman Akbari, Yumei He, Chen Wang, Geng Yuan, Weiwei Chen, and Yanzhi Wang. Phyworld: Physics-faithful world model for video generation, 2026. URL https://arxiv.org/abs/2605.19242.

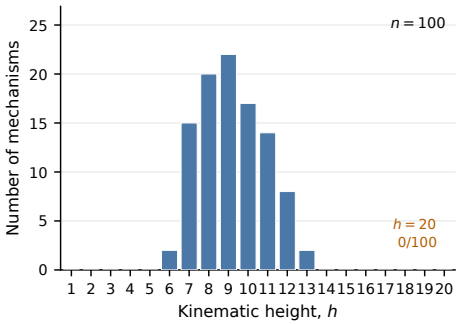
Siyuan Zhou, Hejun Wang, Hu Cheng, Jinxi Li, Dongsheng Wang, Junwei Jiang, Yixiao Jin, Jiayue Huang, Shiwei Mao, Shangjia Liu, Yafei Yang, Hongkang Song, Shenxing Wei, Zihui Zhang, Peng Huang, Shijie Liu, Zhengli Hao, Hao Li, Yitian Li, Wenqi Zhou, Zhihan Zhao, Zongqi He,

Hongtao Wen, Shouwang Huang, Peng Yun, Bowen Cheng, Pok Kazaf Fu, Wai Kit Lai, Jiahao Chen, Kaiyuan Wang, Zhixuan Sun, Ziqi Li, Haochen Hu, Di Zhang, Chun Ho Yuen, Bing Wang, Zhihua Wang, Chuhang Zou, and Bo Yang. Physinone: Visual physics learning and reasoning in one suite. *CVPR*, 2026.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=UdOEZgWJLc.

Tinghui Zhu, Sheng Zhang, James Y. Huang, Selena Song, Xiaofei Wen, Yuankai Li, Hoifung Poon, and Muhao Chen. Video models can reason with verifiable rewards, 2026b. URL https://arxiv.org/abs/2605.15458.

Lamberto Ballan Ziliotto Filippo, Luciano Serafini and Tommaso Campari. What vggt knows about overlap: Probing geometric foundation models for co-visibility. In *European Conference on Computer Vision (ECCV)*, 2026.



(a) Height Distribution via Random Generation

Training	Gear count $N = 200$			
	Random Graph		Linear Graph	
	SSR ↑	$\mathcal{E}_{\text{rmd}}$ ↓	SSR ↑	$\mathcal{E}_{\text{rmd}}$ ↓
Random	97%	0.041	70%	0.279
Height-Centric	98%	0.038	100%	0.027

(b) Quantitative Evaluation on Graph Topologies

Figure 7: **Random vs. kinematic-height-centric dataset generation. (a)** Distribution of kinematic heights across 100 mechanisms generated via random sampling with an exact gear count of $N = 20$. The resulting heights rarely reach the feasible maximum height ($h = 20$). **(b)** Quantitative evaluation reporting Simulation Success Rate (SSR, %) and Relative Motion Discrepancy ($\mathcal{E}_{\text{rmd}}$) on models trained on random vs. height-centric dataset generation strategies. *Top row:* A model trained on random generation pipeline performs poorly on linear chains. *Bottom row:* Training with our height-centric pipeline achieves near-perfect accuracy across both topologies.

A Dataset Generation

This section describes the procedural pipelines used to generate the datasets for this study.

A.1 Random Generation.

We first sample a fixed module m, which remains constant across all gears within a given system. Gears are then added iteratively until the target gear count is reached. For each gear, we sample the number of teeth N uniformly from $[8, 24]$. Given the fixed module m, the pitch diameter is defined as: $D = mN$. The first gear is placed at a random spatial coordinate within the 2D image space. For each subsequent gear, we randomly select an existing gear to serve as its meshing partner. We then sample an angle $\theta \in [0, 2\pi]$ to define the orientation of the axis connecting their centers. Because the distance between the centers of two meshing gears must equal the sum of their pitch radii, the new gear's position is uniquely determined. Finally, we perform a collision check. A placement is successful only if the new gear meshes exclusively with its intended partner without overlapping any other gears in the system; if a collision is detected, the generation step for that specific gear is repeated. Because our study focuses on the topological complexity of the underlying kinematic graph, the specific geometry of the gear teeth is outside the scope of our analysis. We therefore use a standard involute gear profile with a $20°$ pressure angle and a backlash of 0.15 for illustrative purposes. During training, the dataset is generated dynamically with varying gear counts, exposing the model to a diverse range of configurations. For evaluation, we sample 100 random test data for each inference setting using random seeds disjoint from those used during training.

A.2 Kinematic-height centric generation.

A key limitation of the random generation pipeline is that it rarely produces kinematic graphs with large heights, because parent gears are selected uniformly at random. For example, as shown in Fig. 7 (a), when randomly generating 100 mechanisms containing exactly 20 gears, we rarely obtain mechanisms with kinematic height 20, as this requires all 20 gears to form a linear chain (see

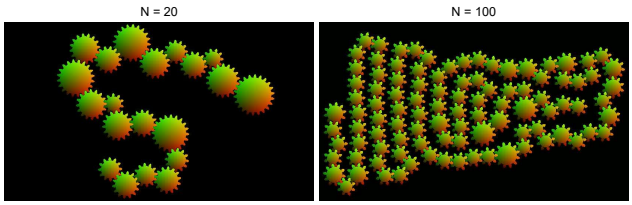


Figure 8: **Examples of linear kinematic graphs.** In a linear gear system, every interior gear meshes with exactly two neighbors, while the two endpoint gears have a degree of one.

Table 4: **Importance of resize augmentation.** Simulation Success Rate (SSR, %) and Relative Motion Discrepancy ($\mathcal{E}_{\text{rmd}}$) for models trained with and without resize augmentation, evaluated on in-distribution (I.D.) ($N = 10, H = 10$) and out-of-distribution (O.O.D.) ($N = 100, H = 10$ and $N = 100, H = 20$) gear mechanisms. Here, N denotes the number of gears and H denotes the height of the kinematic tree. Resize augmentation substantially improves generalization to unseen gear counts at an in-distribution height ($N = 100, H = 10$), while providing no improvement on the O.O.D. height setting ($N = 100, H = 20$), where the success rate remains 0%. These results indicate that resize augmentation is important for studying generalization to unseen gear counts, but is insufficient for generalization to unseen kinematic heights.

	I.D. gear-count and height ($N = 10, H = 10$)		**O.O.D. gear-count, I.D. height** ($N = 100, H = 10$)		**O.O.D. gear-count and height** ($N = 100, H = 20$)	
Strategy	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓
w/o Resize Aug.	100%	0.034	0%	0.932	0%	18.6
w/ Resize Aug.	100%	0.031	83%	0.062	0%	10.1

Fig. 8) and is extremely unlikely under random parent selection. As discussed in the main text, kinematic height is a crucial determinant of task difficulty. Indeed, as shown in the first row of Fig. 7 (b), models trained solely on randomly generated datasets exhibit a significant performance drop on strictly linear kinematic graphs, even when the total gear count remains within the training distribution.

To address this limitation, we modify the dataset generation pipeline. We first uniformly sample a target kinematic height and construct a linear kinematic chain by iteratively attaching new child gears to the terminal end until the target height is reached. The remaining gears are then attached to randomly selected parent gears, subject to the constraint that the overall kinematic height of the tree remains unchanged. As shown in the second row of Fig. 7 (c), training with this height-centric pipeline yields perfect successful rates on linear graphs, as the model is frequently exposed to linear-chain kinematics during training.

A.3 Controlling gear size.

Another critical factor we control is gear size (i.e., radius). When gear configurations are generated naively, the average gear size decreases as the total number of gears increases, in order to fit the entire system within the fixed image boundaries. We empirically find that the video diffusion transformer exhibits poor extrapolation to spatial scales that are out of distribution at inference time. For example, as shown in the first row of Tab. 4, a model trained exclusively on configurations with 10 gears and height 10 exhibits a sharp performance collapse when evaluated on systems with 100 gears and height 10. In this setting, the individual gears are substantially smaller than those seen during training, resulting in a 0% success rate. In contrast, when we introduce resize augmentation during training to expose the model to a wider range of gear sizes, it achieves substantially improved generalization to 100 gears at height 10, as shown in the bottom row.

We therefore apply aggressive spatial resizing augmentations during training. Specifically, we allow training gear sizes to scale down to as small as 12 pixels, while carefully constructing the inference dataset such that all tested gears maintain a minimum size of 16 pixels. This augmentation allows us to evaluate the model's kinematic reasoning capability with minimal confounding from spatial scale.

B Video Encoding Details

We represent each gear $i \in V$ using its normalized object coordinates (NOCs) Wang et al. (2019). As illustrated in Fig. 1, this representation provides dense correspondence for every point on a gear's surface throughout the video. The NOCs $\boldsymbol{r} = [r_x, r_y]^\top \in [-1, 1]^2$ are defined relative to the gear's center $\boldsymbol{c}_i$ and tip radius $R_i = (D_i + 2a)/2$. Specifically, for a pixel at location (x, y) at time t, the normalized coordinates $\boldsymbol{r}$ are given by applying the inverse of the gear's current rotation:

$$\begin{bmatrix} r_x \\ r_y \end{bmatrix} = \frac{1}{R_i} \begin{bmatrix} \cos\theta_i(t) & \sin\theta_i(t) \\ -\sin\theta_i(t) & \cos\theta_i(t) \end{bmatrix} \begin{bmatrix} x - c_{i,x} \\ y - c_{i,y} \end{bmatrix}. \tag{1}$$

We store the r_x and r_y values in the red and green channels, while setting the blue channel and all background pixels (i.e., pixels that do not belong to any gear) to -1.

C VIDEO PARSING DETAILS

Overview. To quantitatively evaluate the motion dynamics of generated videos $\mathbf{V}$, we develop a video parsing pipeline that extracts the gear centers $\mathbf{c}_i$, tip radii R_i, and per-frame rotation angles $\theta_i(t)$ for each gear i. The pipeline consists of two stages.

In the first stage, we identify the individual gears by detecting centers $\boldsymbol{c}_i$ and tip radii R_i from the first frame. This procedure is based on the observation that for any local patch whose pixels $\boldsymbol{p}$ belong to the same gear, the center $\boldsymbol{c}_i$ and tip radius $R_i = D_i/2 + m$ can be recovered by the following rigid transformation:

$$\boldsymbol{p} = R_i \cdot \boldsymbol{n} + \boldsymbol{c}_i$$

where $\boldsymbol{p}$ are image coordinates and $\boldsymbol{n} \in [-1, 1]^2$ are the normalized coordinates. By applying a sliding window and solving a least-squares problem for each window, we can aggregate local estimates of centers and radii, which are then pruned using non-maximum suppression. In the second stage, we compute the per-frame rotation angles $\theta_i(t)$ for each detected gear. Using the known centers $\boldsymbol{c}_i$ and tip radii R_i from the first stage, we can similarly solve a least-squares problem for every frame to compute the exact rotation angle $\theta_i(t)$, as explained in Secs. C.1 to C.3.

Details of each step are explained below.

C.1 DETERMINISTIC LOCAL VOTING FOR GEAR CENTERS

To detect and separate individual gears from the unsegmented foreground, we leverage a technique inspired by the Generalized Hough Transform. We slide an 11×11 local window over the foreground mask. For each window containing a sufficient number of foreground points, we extract a set of normalized coordinates $\{\mathbf{n}_k\} \subset [-1, 1]^2$ and their corresponding spatial locations $\{\mathbf{p}_k\} \subset \mathbb{R}^2$. In an ideal scenario where all foreground pixels within the patch belong to the same gear i, the relationship can be modeled by the rigid transformation $\mathbf{p}_k = A\mathbf{n}_k + \mathbf{t}$, where A is a transformation matrix (representing rotation and scale) and $\mathbf{t}$ represents the center coordinates C_i in the image space. Motivated by this, we solve the ordinary least-squares problem:

$$\min_{A,\mathbf{t}} \sum_k \|\mathbf{p}_k - (A\mathbf{n}_k + \mathbf{t})\|_2^2$$

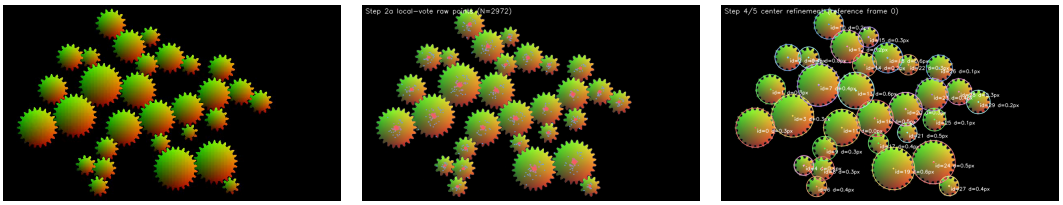
to obtain initial estimates for A and $\mathbf{t}$. However, in practice, the resulting A may be non-orthogonal due to noise in the predicted coordinates or the local window overlapping with several gears. To recover a valid rigid transformation, we follow the principles of Procrustes analysis, where we perform singular value decomposition (SVD) on the estimated matrix, $A = U\Sigma V^T$, which allows us to extract the optimal rotation matrix $\mathcal{R} = UV^T$ and the uniform scale factor $s = \frac{1}{2}\text{Tr}(\Sigma)$. Finally, we refine the local translation (i.e., gear center) estimate $\mathbf{t}$ by computing the median of the residual:

$$\mathbf{t} = \underset{k}{\text{median}}\big(\mathbf{p}_k - s\mathcal{R}\mathbf{n}_k\big)$$

In practice, we perform these operations only on the first frame ($t = 0$), where all normalized coordinate maps are assumed to be at a canonical orientation ($\theta_i(0) = 0$). Consequently, we discard the estimated $\mathcal{R}$ from the subsequent steps. Additionally, we compute a confidence weight $w = \frac{N_{\text{fg}}}{1+\epsilon^2_{\text{med}}}$ for each local estimate $\{\mathbf{t}_m, s_m\}$, where N_{fg} is the number of foreground pixels in the local window and $\epsilon^2_{\text{med}} = \underset{k}{\text{median}}\big(\mathbf{p}_k - s\mathcal{R}\mathbf{n}_k - \mathbf{t}\big)^2$ is the squared median error. As shown in Fig. 9 (b), we observe that local estimates far from the ground-truth center have a significantly smaller confidence score relative to the estimates closer to the ground-truth center, which are utilized to determine the final set of gear centers in the subsequent procedure.

C.2 CENTER CLUSTERING AND RE-ESTIMATION

The local estimates of translation and scale, along with their confidence scores $\{\mathbf{t}_m, s_m, w_m\}$ successfully recover all true gear centers, but also produce duplicate detections and false positives. To



(a) Input Foreground Frame (b) Voting Confidence Heatmap (c) Extracted Centers and Radii

Figure 9: **Visualization of the video parsing pipeline.** **(a)** Initial frame ($t = 0$) with an unsegmented foreground mask. **(b)** Local estimates of gear centers generated via a sliding window. Estimates near true gear centers exhibit high-confidence peaks (red: high confidence, blue: low confidence). **(c)** Final identified gear centers $\mathbf{c}_i$ and tip radii R_i obtained after non-maximum suppression (NMS) and IRLS re-estimation.

obtain the final gear set V, we sort the detected centers in descending order of their confidence scores w, and iteratively add or discard them. Specifically, for each local estimate, we check for spatial overlap: if the circular region defined by its estimated center and scale overlaps significantly with that of a higher-confidence center already included in the final set, the lower-confidence center is discarded. We find this procedure highly effective at removing false positives (See Fig. 9 (c)). Finally, we re-estimate the tip radii $\{R_i\}_{i\in V}$ and final gear centers $\{C_i\}_{i\in V}$ by performing Iteratively Reweighted Least Squares (IRLS) on the pixels close to each detected center, suppressing outliers and eliminating the influence of pixels near gear meshing boundaries.

C.3 Per-frame Motion Angle Estimation

Given the final set of gear centers $\{C_i\}_{i\in V}$ and tip radii $\{R_i\}_{i\in V}$, we extract the orientation of the gears across all frames $t \in [0, 1]$. By substituting the linear coefficients $a = R_i \cos\theta_i(t)$ and $b = R_i \sin\theta_i(t)$, and defining the relative image coordinates (in a y-up convention) as $v_x = x - C_{i,x}$ and $v_y = C_{i,y} - y$, the mapping between the normalized coordinates $\mathbf{n} = [n_x, n_y]^\top$ and relative image coordinates $\mathbf{v} = [v_x, v_y]^\top$ yields the following rigid transformation system:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix} \implies \begin{cases} v_x = an_x - bn_y \\ v_y = bn_x + an_y \end{cases}$$

To estimate (a, b) over a set of N foreground pixels assigned to a gear i, we minimize the squared residual error $E(a, b)$:

$$E(a,b) = \sum_{k=1}^{N} \Big((v_{x,k} - an_{x,k} + bn_{y,k})^2 + (v_{y,k} - bn_{x,k} - an_{y,k})^2 \Big)$$

Taking the partial derivatives with respect to a and b and setting them to zero yields an independent, closed-form ordinary least squares solution for both a and b:

$$a = \frac{\sum_k (v_{x,k} n_{x,k} + v_{y,k} n_{y,k})}{\sum_k (n_{x,k}^2 + n_{y,k}^2)}$$

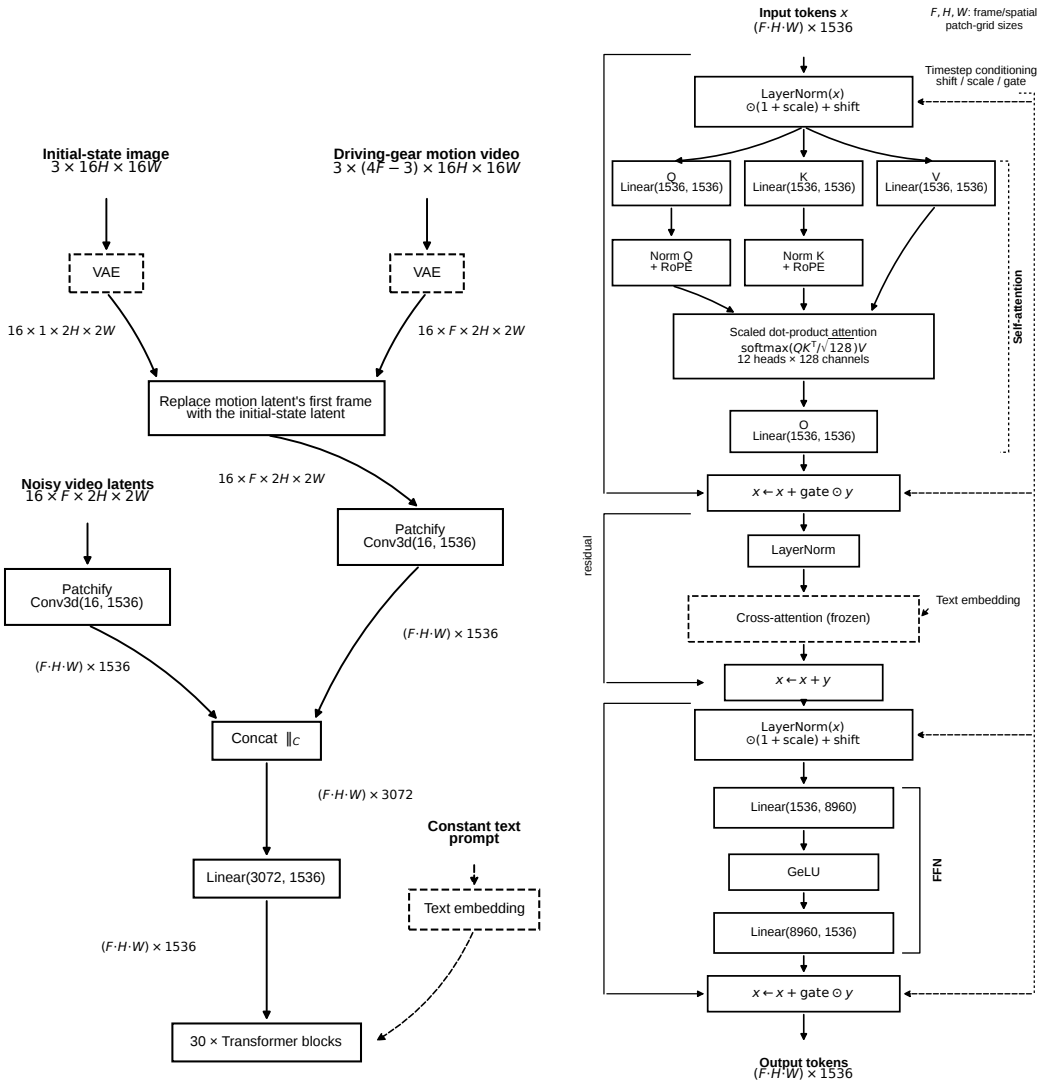
$$b = \frac{\sum_k (v_{y,k} n_{x,k} - v_{x,k} n_{y,k})}{\sum_k (n_{x,k}^2 + n_{y,k}^2)}$$

Once a and b are computed, the rotation angle $\theta_i(t)$ for the frame is recovered:

$$\theta_i(t) = \text{atan2}(b, a)$$

D Architecture and Training Details

We build our architecture upon the open-source Wan2.1-1.3B text-to-video diffusion transformer Wan (2025) based on *Diffsynth Studio* Artiprocher & Team (2024) and finetune the model on



(a) Conditioning & Input Pipeline (b) Transformer Block Architecture

Figure 10: **Architecture details of diffusion transformers (a)** Our conditioning injection mechanism **(b)** A single transformer block of Wan2.1 13B Wan (2025), showing self-attention, cross-attention, and FFN layers. The initial state image and driving-gear motion video are VAE-encoded, concatenated with the noisy video latents along the channel dimension, and projected to the original DiT hidden dimension via a trainable linear projection. Cross-attention modules are kept frozen with a constant text prompt.

our synthetic dataset. The Wan2.1-1.3B model employs a latent diffusion transformer comprising 30 transformer blocks. Each block operates on patchified video latents and is composed of multi-head self-attention, cross-attention, and feed-forward network (FFN) layers. Specifically, within the self-attention module, the inputs are mapped via learnable projections into Query (Q), Key (K), and Value (V) vectors. The model then performs scaled dot-product attention, defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where d_k represents the dimensionality of the key vectors. The result is then projected back into the residual stream via a learnable output projection (O). Following self-attention, the features pass through a cross-attention layer to integrate text prompt embeddings. Finally, the combined repre-

sentations are processed by the FFN—composed of two multi-layer perceptrons (MLPs)—before advancing to the subsequent transformer block (See Fig. 10b for an illustration).

To adapt this model for our gear simulation task, we minimally modify the input architecture to accommodate our specific conditioning signals. Specifically, the model takes two inputs: a first-frame image depicting the initial state of the entire gear configuration, and a single video specifying the motion of the driving gear. Both inputs are encoded through the frozen VAE and concatenated with the patchified video latents along the channel dimension. Notably, we replace the first-frame latent of the driving gear video with the latent representation of the initial state image. To maintain architectural compatibility, a trainable linear layer projects these expanded feature maps back to the model's original channel dimension. Because text prompts are irrelevant to our analysis, we fix the text conditioning to an arbitrary constant prompt during both training and inference. We fine-tune all parameters while keeping the cross-attention modules frozen (see Fig. 10a for an illustration).

We train the model at a spatial resolution of 480×832 with 49 frames. Optimization is performed using AdamW Loshchilov (2017) with a batch size of up to 16, a weight decay of 1×10^{-3}, and a 1,000-step linear warmup. The learning rate is initialized at 1×10^{-5} and decays to 1×10^{-6}. During inference, we employ 50 sampling steps for all tasks, adopting the identical noise scheduling strategy used during training. Notably, we constrain the sampling noise scale to $\sigma \in [1.0, 0.98]$; because our gear simulation task generates predominantly low-frequency video that requires high noise injection to collapse.

We opt to fine-tune a pre-trained video diffusion model rather than training from scratch primarily due to computational constraints. Although the original Wan2.1 checkpoint lacks the capability to reason over kinematic chains, it already possesses a robust prior for denoising and generating visually coherent videos. Fine-tuning allows the network to leverage this prior and focus entirely on learning kinematic reasoning capabilities. Furthermore, as demonstrated by Chao et al. (2026), the resultant reasoning capabilities of fine-tuned models are expected to be commensurate with those of models trained from scratch. Finally, because industiry-standard video diffusion transformers are invariably pre-trained on diverse datasets, our analysis closely reflects how these large-scale foundation models would be adapted for kinematic simulation in more practical scenario.

E Feature Probing details

E.1 Details of MLP probing

Through visual inspection via PCA analysis in Fig. 3, we observed that a specific principal component axis separates the rotational parity of the gears. Notably, while some parities are globally consistent with the driving gear (Fig. 3), others exhibit local consistency (Fig. 6). To quantitatively evaluate this behavior, we construct a MLP probing technique. This probe predicts whether a pair of feature vectors—extracted from a specific transformer layer and corresponding to two distinct gears—belongs to the same rotational parity or not.

Concretely, given intermediate feature maps $f \in \mathbb{R}^{C \times F \times H \times W}$ from FFN outputs, we extract all C-dimensional feature vectors corresponding to a specific gear. Because there is a 16×16 spatial compression factor between the original video resolution and the spatial resolution of the feature maps, we downscale the original coordinates and radii of the gears by this rate. Based on these scaled dimensions, we determine the corresponding spatial locations within the feature maps for each gear. We then spatially average the features for each frame and concatenate then stack them along the channel dimension, resulting in a flattened $F \cdot C$-dimensional feature vector per gear.

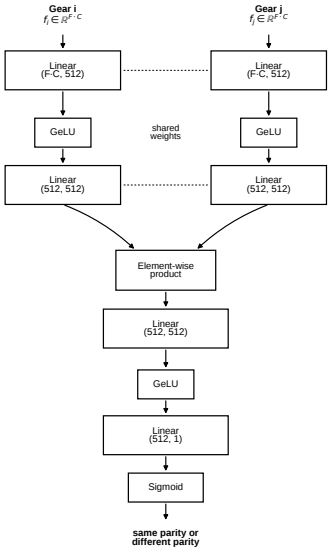
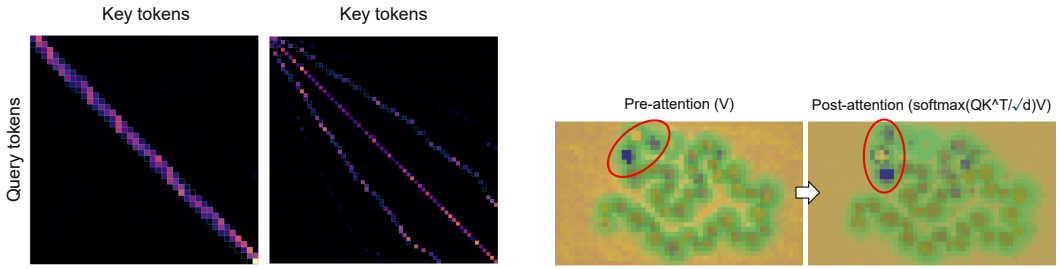


Figure 11: **Architectural details of the MLP probe.** The MLP probe takes two pairs of $(F \cdot C)$-dimensional feature vectors corresponding to two distinct gears and processes them through a weight-shared two-layer MLP, followed by element-wise multiplication ($\odot$) and a second two-layer MLP, and a sigmoid activation to predict whether the two gears share the same rotational parity. GeLU activations are used throughout the hidden layers.



(a) Examples of Node-to-Node attention maps

(b) Joint PCA on pre-/post-attention features

Figure 13: **Self-attention mechanisms propagate parity information to neighboring gears. (a)** Attention maps preferentially attend to adjacent gears, where blue cells highlight adjacent node pairs. **(b)** Joint PCA analysis on pre-self-attention value states (V) versus post-attention outputs shows a one-hop shift in parity representations, demonstrating parity propagation toward neighboring nodes leveraging attention mechanisms.

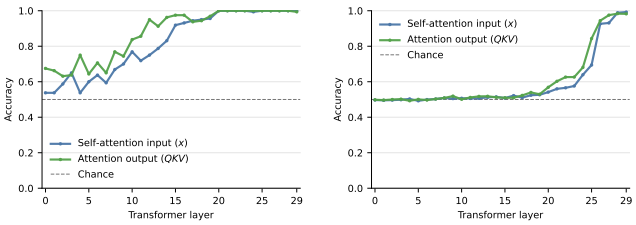
These resultant feature vectors are then processed by learnable MLPs. Specifically, given an arbitrary pair of feature vectors representing two distinct gears, both vectors are first processed independently by a two-layer MLP with shared weights. Their outputs are element-wise multiplied to produce a single 1D vector, which is then further processed by a second two-layer MLP followed by a sigmoid function to yield the prediction. A detailed diagram of this architecture is illustrated in Fig. 11. This symmetric network design is partially inspired by prior work Ziliotto Filippo & Campari (2026); Conneau et al. (2017).

Crucially, we avoid temporally compressing the feature vectors (e.g., via simple temporal average pooling). We found that individual temporal frames do not necessarily encode identical parity information; pooling across the temporal dimension would easily collapse and destroy these distinct cues. We discuss this temporal variation further in Sec. E.3.

The MLP probes are trained using the AdamW optimizer with a learning rate of 10^{-4} and a weight decay of 10^{-4} for 30 epochs. For evaluation, we partition our inference dataset into an 80-sequence training split to train the probes and a 20-sequence test split to demonstrate feature probing results.

E.2 Feature Probing on Self-attention Modules

Self-attention mechanisms play a key role in parity determination. We next investigate which internal component of the transformer blocks is responsible for this progressive propagation. In Fig. 12, we perform feature probing at the inputs and outputs of the self-attention sublayers. We observe a notable increase in prediction accuracy immediately following the self-attention operations. This indicates that the attention operation plays a key role in determining the parity of gears whose parity has not yet been resolved.



(a) Trained on $N \leq 5$
Tested on $N = h = 5$

(b) Trained on $N \leq 40$
Tested on $N = h = 40$

Figure 12: **Self-attention mechanisms play a key role in parity determination.** MLP probing accuracy at self-attention input vs. output shows an increase following scaled dot-product attention for models trained on up to $N = 5$ (**a**) and $N = 40$ (**b**) gears, tested on kinematic chains of height $h = 5$ and $h = 40$, respectively.

Self-attention mechanism propagates parity information to neighbouring gears. In our short-kinematic-chain experiments (Sec. 5.1), attention maps (Fig. 13) reveal heads that preferentially attend to adjacent gears. Furthermore, joint PCA analysis of the inputs and outputs of the scaled dot-product attention operation shows that parity representations shift toward their neighboring nodes.

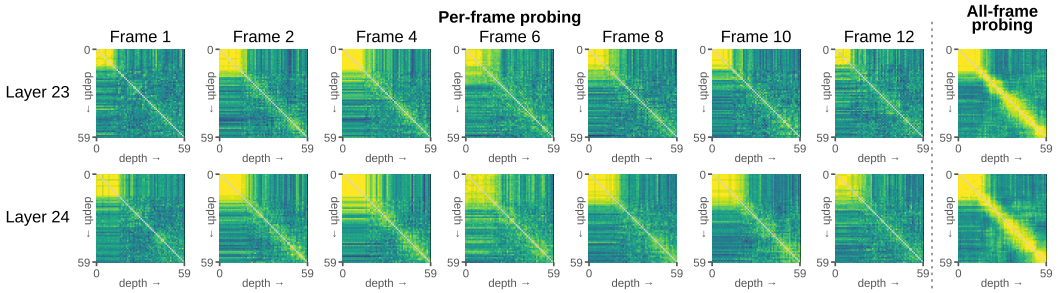


Figure 15: **Per-frame vs. all-frame concatenated feature probing.** Per-frame probing exhibits high prediction variance across frames and weaker accuracy within local kinematic neighborhoods. Conversely, all-frame concatenated probing demonstrates strong accuracy and local consistency, suggesting that video diffusion transformers utilize the temporal dimension to encode non-identical parity representations.

Together, these results suggest that self-attention progressively extends parity predictions across the kinematic tree.

E.3 Temporal variation

A non-trivial aspect of feature probing is how to handle the temporal dimension, whose role in kinematic reasoning is not immediately clear. In particular, determining rotational parity is fundamentally a spatial reasoning problem and, in principle, should not require temporal dimension.

Interestingly, we find that feature maps from different frames do not encode identical parity information. To examine this effect, rather than concatenating feature vectors across all frames, we feed the feature vector from each frame independently into the MLP probe and measure how parity prediction varies across the temporal dimension.

As illustrated in Fig. 15, we observe significant variance in prediction accuracy across frames, especially in deeper layers where locally consistent parity clusters are formed in various ways. This suggests that video diffusion transformers may leverage their temporal dimension to store different states of locally consistent parities during their internal reasoning processes.

Indeed, we initially attempted to average the feature vectors across the temporal dimension; however, as shown in Fig. 14, this resulted in a drop in prediction accuracy around layers 15–25. This degradation indicates that certain parity cues exist only within specific temporal frames. We leave a more rigorous analysis of this temporal variation to future work,

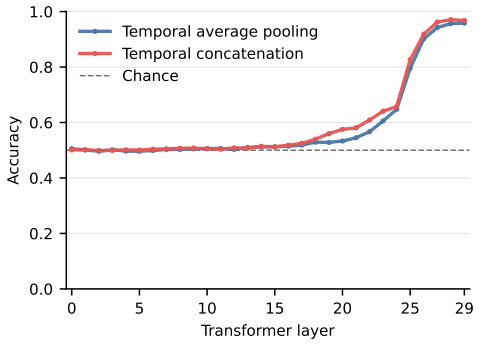


Figure 14: **Temporal average pooling vs. concatenation in feature probing.** Probing accuracy comparison between temporally average-pooled ($\mathbb{R}^F$) and temporally concatenated ($\mathbb{R}^{F\cdot C}$) feature vectors. Average pooling across frames induces a performance drop around layers 15–25, indicating that parity representations are not uniformly encoded across frames in late-middle layers. Models are trained on up to $N = 60$ gears and evaluated on kinematic chains containing 60 gears.

F Analysis on inference steps

While our gear simulation task is inherently deterministic, with the state of the mechanism uniquely defined, we intentionally employ a diffusion formulation. This design allows us to investigate the kinematic reasoning capabilities of standard video diffusion transformers trained with standard diffusion objectives. As shown in Tab. 5, the trained model achieves near-perfect accuracy on single-step pre-

Table 5: **Impact of inference steps on simulation accuracy.** We evaluate model performance across varying numbers of denoising steps (1, 2, 3, 4, 5, 10, and 50). We report the Simulation Success Rate (SSR, %) and relative motion discrepancy ($\mathcal{E}_{\text{rmd}}$) across five settings, where the model is trained on up to gear counts of $N = 10, 20, 30, 40, 50$, and evaluated on linear kinematic chains whose height (h) matches the maximum training gear count. A single-step prediction is sufficient to achieve near-perfect performance, while iterative multi-step sampling yields marginal improvements, potentially due to the self-correcting capabilities of the diffusion process on slightly undertrained models.

	$N = 10, h = 10$		$N = 20, h = 20$		$N = 30, h = 30$		$N = 40, h = 40$		$N = 50, h = 50$	
Steps	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓	**SSR (%)** ↑	$\mathcal{E}_{\text{rmd}}$ ↓
1	99%	0.034	99%	0.029	94%	0.037	92%	0.038	89%	0.169
2	100%	0.033	100%	0.028	98%	0.031	94%	0.040	92%	0.052
3	100%	0.033	100%	0.028	98%	0.036	94%	0.048	95%	0.059
4	100%	0.033	100%	0.028	98%	0.040	96%	0.072	95%	0.231
5	100%	0.033	99%	0.028	98%	0.046	97%	0.091	95%	0.065
10	99%	0.033	100%	0.028	99%	0.036	94%	0.108	94%	0.041
50	100%	0.034	100%	0.028	97%	0.038	97%	0.052	96%	0.040

dictions for in-distribution data, indicating that a single feed-forward computation has sufficient capacity to simulate the underlying gear mechanisms.

However, we occasionally observe marginal improvements in accuracy with multi-step iterative sampling. We hypothesize that this phenomenon stems from two factors: (1) the model is trained across a range of noise levels, whereas one-step prediction only evaluates the highest noise level, at which the model may not have fully converged, and (2) the iterative denoising process may exert a self-correcting mechanism that pulls slightly deviated denoising trajectories back toward the correct manifold Jang et al. (2026). Similar marginal benefits from iterative sampling on deterministic tasks have been reported in prior work Chao et al. (2026).

G Full Quantitative Results for Gear Counts v.s. Kinematic Heights

This section provides the complete evaluation matrices for models trained on gear mechanisms up to $N_{\text{train}} \in \{10, 20, 30, 40, 50, 100\}$ gears. Each table measures simulation success rates (%) and Relative Motion Disparity ($\mathcal{E}_{\text{rmd}}$) evaluated across gear counts $N \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and kinematic heights $h \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

In-distribution configurations ($N \leq N_{\text{train}}$ and $h \leq N_{\text{train}}$) are highlighted in **red**.

G.1 Simulation success rates (%)

Please refer to Tab. Tab. 7, Tab. 8, Tab. 9, Tab. 10, Tab. 11, Tab. 12.

G.2 Relative Motion Disparity ($\mathcal{E}_{\text{RMD}}$)

Please refer to Tab. Tab. 14, Tab. 15, Tab. 16, Tab. 17, Tab. 18, Tab. 19.

H Full MLP Probing Accuracy Across Layers and Kinematic Depths

This section presents the complete MLP probing results for predicting the parity of each gear relative to the driving gear. We visualize the probing accuracy across transformer layers versus kinematic depth for models trained on maximum gear counts of $N \in \{5, 10, 20, 30, 40, 50\}$, as detailed in Fig. 16, Fig. 17, Fig. 18, Fig. 19, Fig. 22 Fig. 21, respectively. These plots demonstrate how parity

representations for nodes at varying kinematic depths are progressively resolved across the layers of the network.

I Full Per-Layer Pairwise Parity Prediction Accuracy

This section presents the full pairwise parity prediction accuracy across transformer layers for models trained on gear counts up to 5, 10, 20, 30, 40, and 50 gears, as detailed in Fig. 23, Fig. 24, Fig. 25, Fig. 26, Fig. 27 Fig. 28, respectively. The visualized matrices demonstrate how the model progressively resolves parity across transformer layers.

Table 6: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 5$)**.**

	Kinematic Height (h)				
Gears (N)	**5**	**10**	**20**	**30**	**40**
5	100%	—	—	—	—
10	100%	3%	—	—	—
20	94%	0%	0%	—	—
30	96%	0%	0%	0%	—
40	88%	0%	0%	0%	0%

Table 7: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 10$)**.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	100%	—	—	—	—	—	—	—	—	—
20	100%	1%	—	—	—	—	—	—	—	—
30	97%	1%	0%	—	—	—	—	—	—	—
40	100%	0%	0%	0%	—	—	—	—	—	—
50	94%	0%	0%	0%	0%	—	—	—	—	—
60	90%	0%	0%	0%	0%	0%	—	—	—	—
70	100%	0%	0%	0%	0%	0%	0%	—	—	—
80	99%	0%	0%	0%	0%	0%	0%	0%	—	—
90	89%	0%	0%	0%	0%	0%	0%	0%	0%	—
100	82%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 8: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 20$)**.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	100%	—	—	—	—	—	—	—	—	—
20	100%	100%	—	—	—	—	—	—	—	—
30	100%	99%	68%	—	—	—	—	—	—	—
40	99%	98%	63%	42%	—	—	—	—	—	—
50	98%	86%	49%	23%	12%	—	—	—	—	—
60	98%	80%	33%	18%	9%	6%	—	—	—	—
70	100%	70%	22%	16%	12%	7%	2%	—	—	—
80	100%	54%	19%	7%	10%	3%	2%	3%	—	—
90	100%	46%	10%	5%	4%	7%	2%	3%	0%	—
100	97%	28%	15%	4%	5%	1%	2%	1%	3%	4%

Table 9: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 30$)**.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	100%	—	—	—	—	—	—	—	—	—
20	100%	99%	—	—	—	—	—	—	—	—
30	99%	99%	97%	—	—	—	—	—	—	—
40	99%	99%	94%	84%	—	—	—	—	—	—
50	99%	98%	86%	74%	58%	—	—	—	—	—
60	97%	88%	78%	66%	47%	32%	—	—	—	—
70	100%	80%	72%	37%	34%	30%	14%	—	—	—
80	100%	70%	59%	31%	27%	16%	11%	12%	—	—
90	100%	68%	47%	24%	24%	12%	16%	5%	2%	—
100	98%	53%	38%	24%	15%	7%	7%	3%	5%	3%

Table 10: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 40$)**.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	99%	—	—	—	—	—	—	—	—	—
20	99%	99%	—	—	—	—	—	—	—	—
30	99%	100%	100%	—	—	—	—	—	—	—
40	100%	99%	100%	97%	—	—	—	—	—	—
50	99%	97%	98%	96%	84%	—	—	—	—	—
60	98%	90%	88%	86%	66%	54%	—	—	—	—
70	100%	96%	81%	60%	44%	47%	30%	—	—	—
80	100%	92%	78%	58%	46%	31%	25%	17%	—	—
90	100%	83%	69%	51%	43%	31%	23%	20%	10%	—
100	99%	84%	71%	40%	35%	16%	12%	10%	9%	6%

Table 11: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 50$)**.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	100%	—	—	—	—	—	—	—	—	—
20	100%	100%	—	—	—	—	—	—	—	—
30	99%	100%	100%	—	—	—	—	—	—	—
40	99%	100%	100%	100%	—	—	—	—	—	—
50	99%	99%	97%	98%	95%	—	—	—	—	—
60	99%	95%	97%	96%	95%	85%	—	—	—	—
70	100%	99%	89%	67%	56%	54%	43%	—	—	—
80	100%	97%	84%	62%	57%	42%	35%	31%	—	—
90	100%	90%	74%	63%	46%	38%	29%	25%	19%	—
100	99%	92%	73%	47%	35%	20%	23%	13%	15%	5%

Table 12: **Simulation success rates** (%) ($N_{\text{train}}, h_{\text{train}} \leq 100$)**.**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	99%	—	—	—	—	—	—	—	—	—
20	100%	98%	—	—	—	—	—	—	—	—
30	99%	100%	100%	—	—	—	—	—	—	—
40	100%	99%	99%	100%	—	—	—	—	—	—
50	99%	96%	97%	95%	97%	—	—	—	—	—
60	96%	95%	92%	97%	98%	96%	—	—	—	—
70	100%	99%	97%	97%	93%	91%	98%	—	—	—
80	99%	100%	98%	92%	93%	94%	86%	99%	—	—
90	100%	100%	96%	94%	90%	84%	90%	84%	98%	—
100	100%	99%	95%	93%	90%	87%	83%	87%	86%	89%

Table 13: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 5$).**

	Kinematic Height (h)				
Gears (N)	**5**	**10**	**20**	**30**	**40**
5	0.048	—	—	—	—
10	0.04	2.699	—	—	—
20	0.077	2.352	3.161	—	—
30	0.044	1.931	2.72	2.651	—
40	0.052	2.109	3.493	3.562	5.053

Table 14: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 10$).**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.034	—	—	—	—	—	—	—	—	—
20	0.03	2.855	—	—	—	—	—	—	—	—
30	0.043	3.993	6.742	—	—	—	—	—	—	—
40	0.031	4.382	7.37	12.15	—	—	—	—	—	—
50	0.042	5.484	7.243	8.997	11.29	—	—	—	—	—
60	0.05	5.522	8.161	10.01	9.97	7.329	—	—	—	—
70	0.031	9.431	19.41	19.48	14.84	16.89	17.18	—	—	—
80	0.033	9.229	20.98	19.87	14.2	16.86	13.44	14.01	—	—
90	0.041	13.49	17.27	21.23	21.52	18.3	19.21	15.84	18.7	—
100	0.067	10.11	25.3	28.78	27.13	23.27	29.12	27.88	21.76	20.45

Table 15: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 20$).**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.032		—	—	—	—	—	—	—	—
20	0.026	0.028	—	—	—	—	—	—	—	—
30	0.026	0.027	1.675	—	—	—	—	—	—	—
40	0.027	0.042	0.64	3.545	—	—	—	—	—	—
50	0.034	0.095	1.185	4.43	6.86	—	—	—	—	—
60	0.038	0.335	2.068	2.45	5.745	5.99	—	—	—	—
70	0.027	1.108	4.3	4.309	6.098	8.819	10.76	—	—	—
80	0.028	0.727	5.995	7.608	7.337	11.4	7.632	8.81	—	—
90	0.032	1.497	3.539	8.148	10.94	6.151	11.44	10.21	13.63	—
100	0.046	0.943	7.78	6.679	8.032	11.11	14.84	16.16	11.39	12.87

Table 16: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 30$).**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.033		—	—	—	—	—	—	—	—
20	0.027	0.029		—	—	—	—	—	—	—
30	0.029	0.028	0.038	—	—	—	—	—	—	—
40	0.028	0.027	0.034	0.295	—	—	—	—	—	—
50	0.035	0.035	0.083	0.361	1.01	—	—	—	—	—
60	0.037	0.091	0.128	0.543	1.304	2.278	—	—	—	—
70	0.025	0.176	0.27	1.325	1.719	2.049	3.389	—	—	—
80	0.026	0.277	0.456	1.834	2.638	3.448	3.677	4.097	—	—
90	0.029	0.307	0.72	3.209	3.62	2.898	5.621	4.507	6.513	—
100	0.039	0.632	2.032	3.44	3.935	5.426	7.787	7.845	6.702	7.203

Table 17: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 40$).**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.045		—	—	—	—	—	—	—	—
20	0.037	0.029		—	—	—	—	—	—	—
30	0.028	0.028	0.028	—	—	—	—	—	—	—
40	0.027	0.027	0.027	0.052	—	—	—	—	—	—
50	0.033	0.034	0.035	0.059	0.201	—	—	—	—	—
60	0.036	0.037	0.056	0.114	0.945	0.658	—	—	—	—
70	0.025	0.068	0.208	0.63	0.794	2.049	2.006	—	—	—
80	0.026	0.058	0.242	0.717	1.077	2.238	2.11	2.829	—	—
90	0.028	0.109	0.28	1.352	2.928	1.523	3.285	2.658	4.14	—
100	0.039	0.219	0.647	1.477	1.95	3.582	4.51	4.842	3.836	4.673

Table 18: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 50$).**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.032		—	—	—	—	—	—	—	—
20	0.027	0.028		—	—	—	—	—	—	—
30	0.027	0.027	0.027		—	—	—	—	—	—
40	0.026	0.025	0.026	0.028		—	—	—	—	—
50	0.032	0.032	0.033	0.034	0.052	—	—	—	—	—
60	0.035	0.033	0.037	0.036	0.054	0.182	—	—	—	—
70	0.024	0.029	0.197	0.443	0.558	0.799	0.898	—	—	—
80	0.024	0.028	0.266	0.603	0.426	1.682	1.595	1.704	—	—
90	0.027	0.088	0.127	1.375	1.34	1.326	2.4	2.323	3.59	—
100	0.036	0.05	0.39	0.661	1.347	3.036	3.189	4.73	3.991	4.361

Table 19: **Relative motion disparity ($\mathcal{E}_{\text{rmd}}$) ($N_{\text{train}}, h_{\text{train}} \leq 100$).**

	Kinematic Height (h)									
Gears (N)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	0.04		—	—	—	—	—	—	—	—
20	0.032	0.037		—	—	—	—	—	—	—
30	0.035	0.034	0.036		—	—	—	—	—	—
40	0.032	0.031	0.032	0.034		—	—	—	—	—
50	0.037	0.04	0.041	0.043	0.041		—	—	—	—
60	0.04	0.052	0.045	0.047	0.042	0.045		—	—	—
70	0.032	0.035	0.042	0.044	0.075	0.12	0.036		—	—
80	0.036	0.033	0.054	0.045	0.107	0.059	0.085	0.036		—
90	0.035	0.034	0.055	0.041	0.121	0.432	0.091	0.128	0.037	
100	0.035	0.039	0.038	0.088	0.081	0.285	1.14	0.294	0.068	0.14

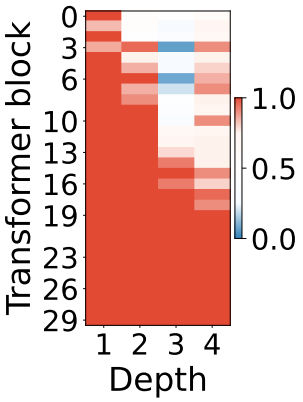


Figure 16: **MLP probing ($N \leq 5$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 5$).

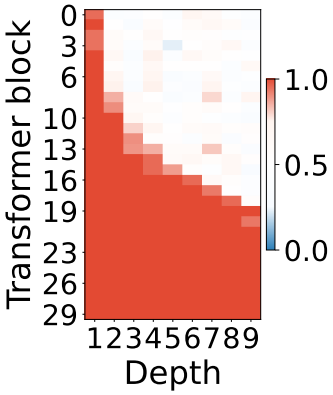


Figure 17: **MLP probing ($N \leq 10$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 10$).

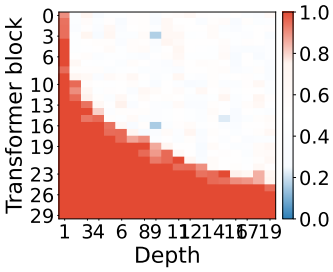


Figure 18: **MLP probing ($N \leq 20$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 20$).

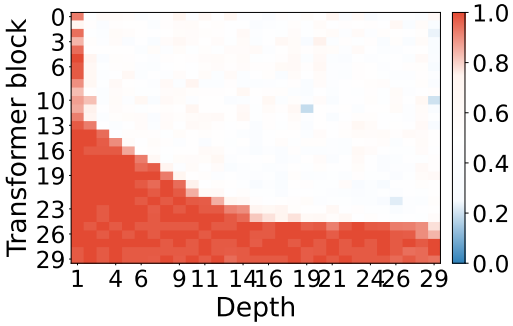


Figure 19: **MLP probing ($N \leq 30$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 30$).

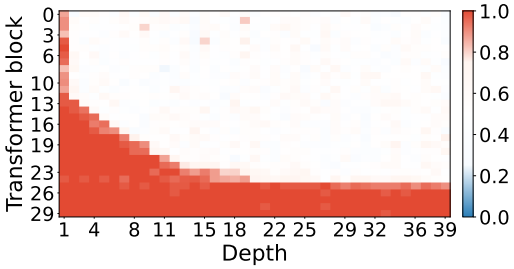


Figure 20: **MLP probing ($N \leq 40$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 40$).

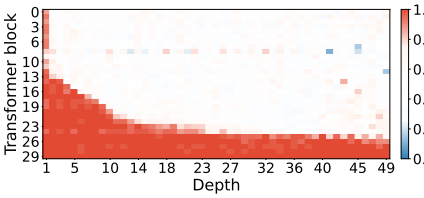


Figure 21: **MLP probing ($N \leq 50$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 50$).

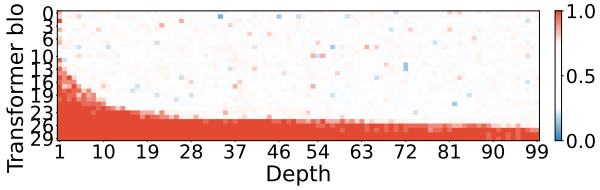


Figure 22: **MLP probing ($N \leq 100$).** Parity accuracy relative to driving gear across layers vs. depth, tested on kinematic chain ($h = 100$).

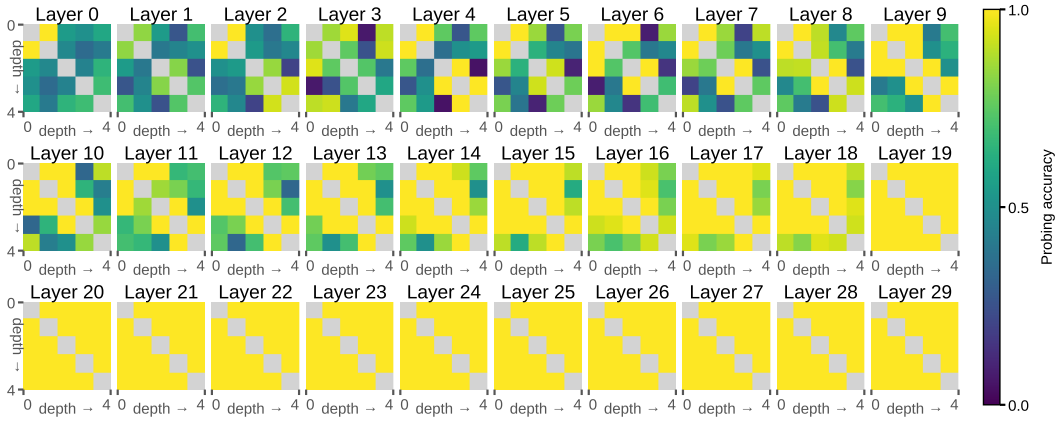


Figure 23: **Pairwise parity prediction accuracy ($N \leq 5$).** Full per-layer accuracy matrices for models trained on up to 5 gears and tested on kinematic chains of height $h = 5$.

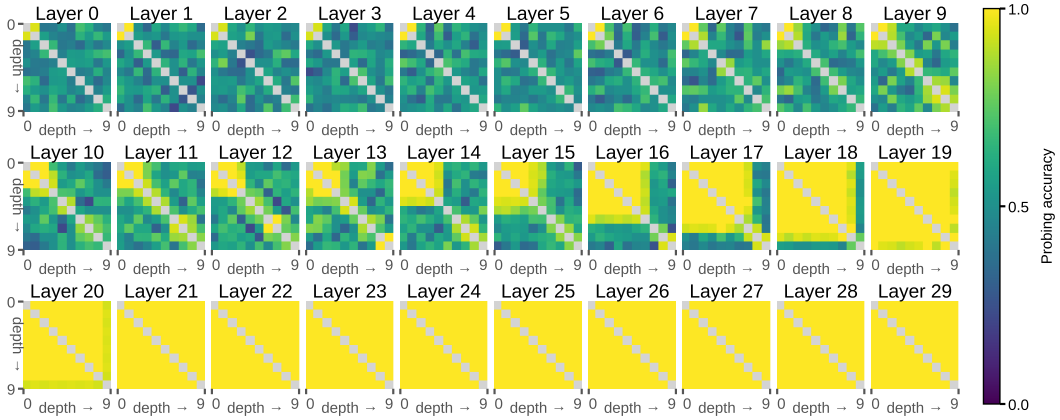


Figure 24: **Pairwise parity prediction accuracy ($N \leq 10$).** Full per-layer accuracy matrices for models trained on up to 10 gears and tested on kinematic chains of height $h = 5$.

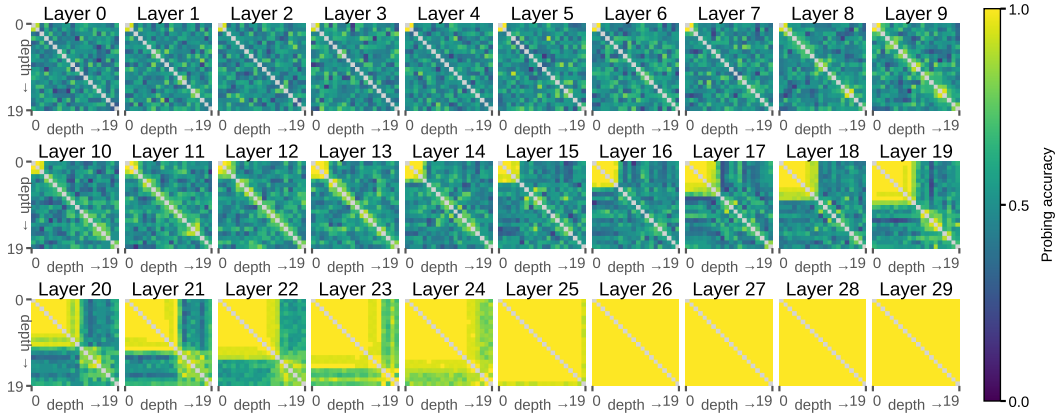


Figure 25: **Pairwise parity prediction accuracy ($N \leq 20$).** Full per-layer accuracy matrices for models trained on up to 5 gears and tested on kinematic chains of height $h = 20$.

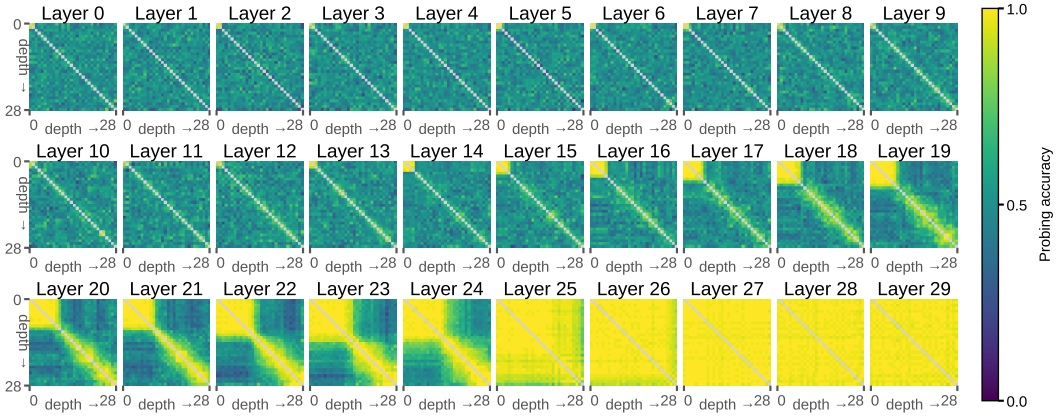


Figure 26: **Pairwise parity prediction accuracy** ($N \leq 30$)**.** Full per-layer accuracy matrices for models trained on up to 30 gears and tested on kinematic chains of height $h = 30$.

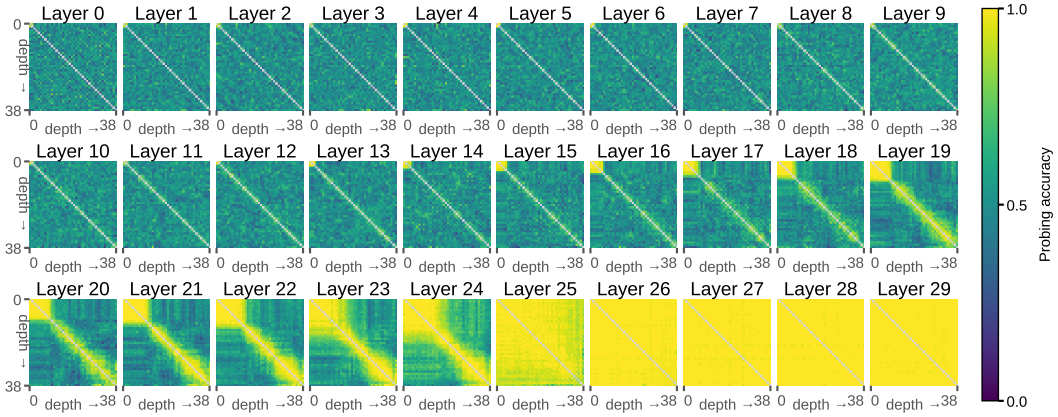


Figure 27: **Pairwise parity prediction accuracy** ($N \leq 40$)**.** Full per-layer accuracy matrices for models trained on up to 5 gears and tested on kinematic chains of height $h = 40$.

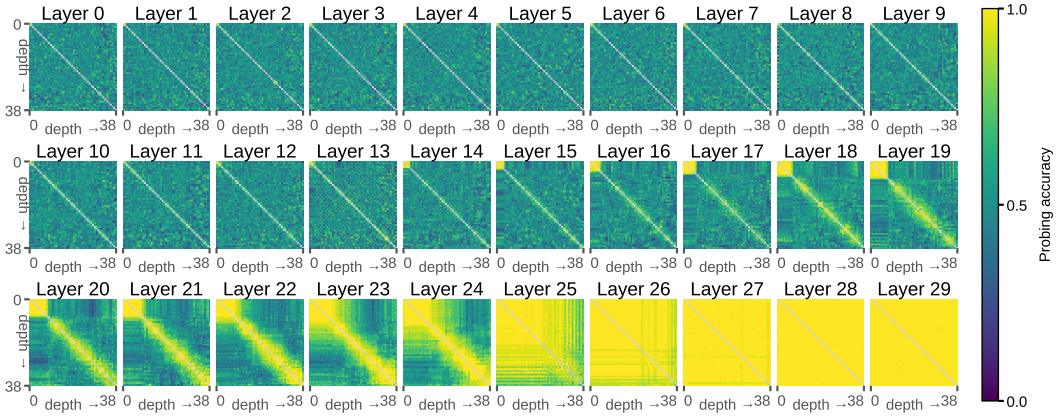


Figure 28: **Pairwise parity prediction accuracy** ($N \leq 50$)**.** Full per-layer accuracy matrices for models trained on up to 50 gears and tested on kinematic chains of height $h = 50$.

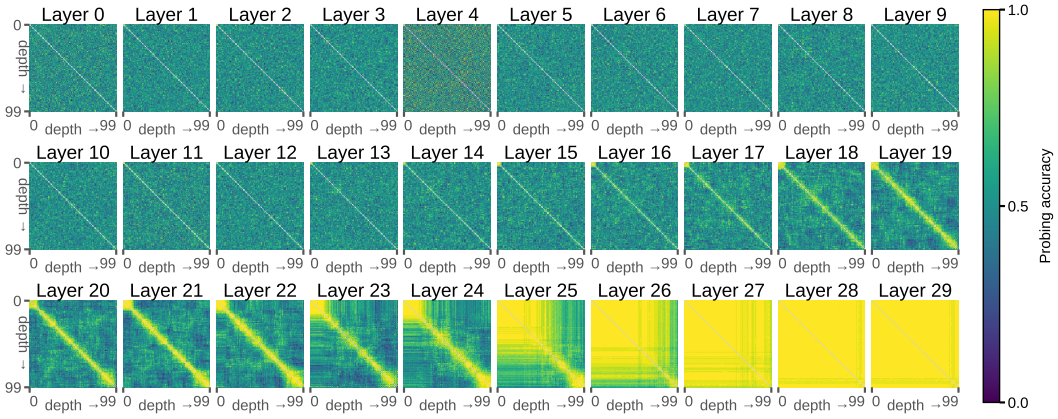


Figure 29: **Pairwise parity prediction accuracy ($N \leq 100$).** Full per-layer accuracy matrices for models trained on up to 100 gears and tested on kinematic chains of height $h = 100$.