PARE: Pruning and Adaptive Routing for Efficient Video Generation

Yutong Wang[1], Yunke Wang[1], Tianfan Xue[3,2], Yu Qiao[2], Yaohui Wang[2], Xinyuan Chen[2]*, Chang Xu[1]*

[1]School of Computer Science, The University of Sydney, Australia
[2]Shanghai Artificial Intelligence Laboratory, China
[3]Multimedia Laboratory, The Chinese University of Hong Kong, China

Abstract

Video Diffusion Transformers (DiTs) generate high-quality videos but demand substantial compute due to wide blocks, deep architectures, and iterative sampling. Recent methods reduce cost by compressing width, depth, or sampling steps, but typically commit to a fixed architecture that cannot adapt to individual inputs or denoising stages. We propose **PARE** (**P**runing and **A**daptive **R**outing for **E**fficient video generation), which jointly compresses width and depth with structure-aware pruning and input-adaptive routing. For width, we observe that attention heads specialize into spatial and temporal roles, and design importance scoring that accounts for this distinction to prevent motion-critical temporal heads from being pruned prematurely. For depth, we train a lightweight router conditioned on denoising timestep and visual content to dynamically select which blocks to execute at each step, enabling per-input compute adaptation rather than static block removal. A progressive pipeline first recovers width-pruned quality via distillation, then jointly optimizes the student and router to decouple the two learning objectives. Experiments on Wan2.1-14B for both image-to-video and text-to-video generation show that PARE substantially reduces per-step computation while preserving quality across VBench dimensions, and composes with step distillation for further acceleration. Code and models are available at `https://github.com/hhhh1138/PARE`.

1 Introduction

Diffusion Transformers [28] have brought video generation to remarkable fidelity [36, 51, 17, 11, 18, 55], yet their inference cost remains prohibitive. Each denoising step must traverse dozens of transformer blocks with billions of parameters, and generating a single video requires tens of such steps. This cost spans three axes: *width*, the heads and FFN dimensions within each block; *depth*, the number of blocks executed per step; and *steps*, the denoising iterations. Recent methods have begun to compress multiple axes jointly [6, 16, 29, 43], yet they remain *static*: the same reduced architecture serves every input at every timestep, all attention heads are treated uniformly regardless of their functional role, and there is no mechanism to adapt computation to input complexity.

Two properties of video DiTs suggest a better strategy. First, attention heads exhibit clear spatial-temporal specialization: shallow heads attend predominantly within frames while deeper heads capture cross-frame motion. Magnitude-based pruning systematically undervalues temporal heads, whose activations are smaller yet critical for motion coherence. Second, block importance varies across both timesteps and inputs: early denoising steps that recover global structure rely on different blocks than late steps that refine details, and the optimal block subset further depends on scene

*Corresponding author.

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

complexity. These observations call for *spatial-temporal aware width pruning* within blocks and *content-adaptive depth routing* across blocks.

We present **PARE** (**P**runing and **A**daptive **R**outing for **E**fficient video generation), a framework that compresses video DiTs along both axes. For width, PARE scores attention heads via calibration-based importance with a protection multiplier for temporal heads, and selects FFN neurons through importance-plus-diversity filtering that exploits the factored up/down projection structure. For depth, a lightweight router conditioned on both timestep and visual content selects which blocks to execute, with a budget schedule that allocates more blocks to high-noise steps. A progressive training pipeline first recovers width-pruned quality via distillation, then jointly optimizes the pruned model and router, avoiding the compound difficulty of learning both from scratch.

We evaluate PARE on Wan2.1-14B for both image-to-video and text-to-video generation. PARE halves per-step computation while preserving quality across VBench [15] dimensions, and composes with step distillation for further acceleration along the third axis. Our contributions are:

- A spatial-temporal aware width pruning strategy combining temporal head protection with importance-plus-diversity FFN selection that exploits the factored projection structure.
- A content-adaptive block router that conditions on both timestep and visual input to select which blocks to execute, with a budget schedule that allocates more computation to high-noise steps, enabling per-sample depth adaptation at negligible parameter overhead.
- A progressive training pipeline that decouples width distillation from joint width-routing optimization, breaking the circular dependency between pruned feature quality and routing decisions.
- Experiments on Wan2.1-14B showing that PARE halves per-step computation with minimal quality loss on both I2V and T2V, and up to $\sim 50\times$ total speedup with step distillation.

2 Related Work

Efficient Video Generation. Efficient video generation methods reduce inference cost through numerical precision [60, 20, 37], attention sparsity [44, 49, 22, 19], model compression [29, 16], and sampling acceleration [53, 52, 39, 38, 4, 41]. Quantization lowers the precision of weights and activations: ViDiT-Q [60] studies quantization for image and video diffusion transformers, while SVDQuant [20] uses a high-precision low-rank branch to absorb outliers and enable 4-bit diffusion inference. Sparse attention reduces token interactions within each denoising step; Sparse VideoGen [44] exploits spatial-temporal sparsity in video attention, and Sparse VideoGen2 [49] improves sparse computation through semantic-aware permutation. Structural pruning reduces network size by removing redundant components, while consistency models [31, 25] and distribution matching distillation [53, 52, 39, 38, 8, 7] reduce the number of sampling steps. These approaches address different sources of inference cost. PARE focuses on reducing per-step computation through width pruning and adaptive depth routing, and combines this compression with step distillation for further acceleration.

Structural Pruning for Diffusion Models. Structural pruning [6, 16, 43, 12, 56, 29] compresses pretrained networks by removing channels, attention heads, FFN dimensions, or entire blocks. For diffusion models, Diff-Pruning [6] aggregates Taylor-based importance estimates over selected timesteps to identify removable parameters. Recent video generation methods combine structural compression with distillation to recover quality. NeoDragon [16] prunes MMDiT blocks according to their relative importance and restores performance through two-stage distillation. MobileDiT [43] introduces sensitivity-aware tri-level pruning together with adversarial step distillation, while FastLightGen [29] jointly addresses model-size and sampling-step reduction. These studies demonstrate the effectiveness of combining structural pruning with distillation. PARE further explores which intra-block components to retain, combining spatial-temporal aware head pruning with FFN selection.

Dynamic Computation for Diffusion Models. Dynamic computation [61, 27, 57, 24, 63] allocates resources according to the requirements of individual inputs or denoising stages. DyDiT [61] explores this direction through timestep-wise dynamic width and spatial-wise dynamic token selection, adapting computation within image DiTs. A complementary line of work exploits redundancy across denoising steps through feature reuse. DeepCache [27] reuses high-level features across adjacent steps, while PAB [63] reuses attention outputs over subsequent steps in video generation.

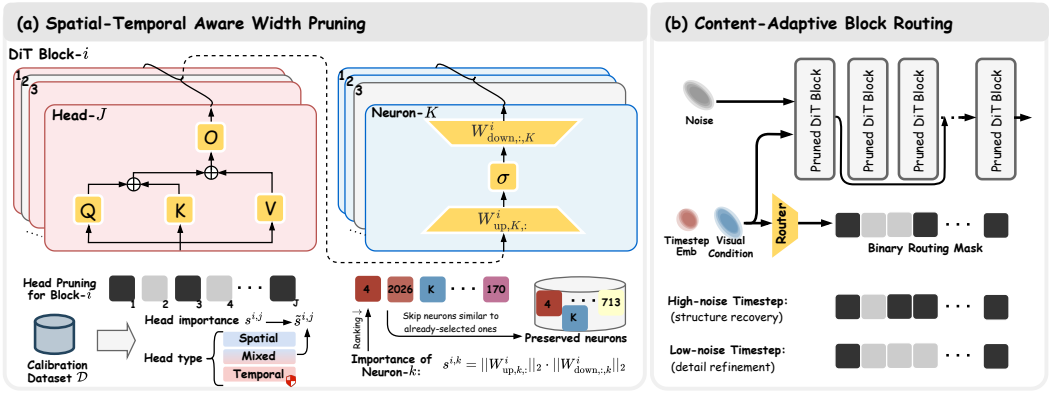

Figure 1: **Overview of PARE. (a)** Spatial-temporal aware width pruning. Calibration-based head importance scores $s^{i,j}$ are adjusted to $\tilde{s}^{i,j}$ according to head type, protecting temporal heads from magnitude bias. FFN neurons are ranked by the product of their up/down projection norms and greedily selected to preserve importance while avoiding functional redundancy. **(b)** Content-adaptive block routing. A lightweight router conditioned on the timestep embedding and visual input produces a binary mask selecting which pruned blocks to execute. A timestep-dependent budget assigns more blocks to high-noise steps for structure recovery and fewer to low-noise steps for detail refinement.

Routing and caching therefore address different aspects of redundant computation: routing selects the components to execute, whereas caching reuses previously computed representations. PARE performs dynamic selection at block granularity within a width-pruned video DiT. A timestep-dependent budget determines how many blocks are active, and a router conditioned on both timestep and visual content determines which blocks to execute. This design enables per-sample depth adaptation within a single compressed model.

3 Method

As illustrated in Figure 1, PARE compresses a video Diffusion Transformer (DiT) [28] along two axes, *width* (intra-block) and *depth* (inter-block), via spatial-temporal aware width pruning (§3.2), content-adaptive block routing (§3.3), and a progressive training pipeline (§3.4).

3.1 Preliminaries

We build on the flow matching framework [21, 23] for video generation. Given a video latent $x_0 \in \mathbb{R}^{F_l \times H_l \times W_l \times C}$ obtained via 3D VAE compression (F_l temporal slices, spatial resolution $H_l \times W_l$, C channels), the forward process constructs a noisy sample $x_t = (1-\sigma_t)x_0 + \sigma_t \epsilon$ at time $t \in [0,1]$, where σ_t increases monotonically from 0 to 1. A velocity network v_Θ learns to predict the flow target $(\epsilon - x_0)$ via:

$$\mathcal{L}_{\text{FM}}(\Theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\|v_\Theta(x_t,t,c) - (\epsilon - x_0)\|_2^2\right], \tag{1}$$

where c denotes the conditioning signal (text, reference image, or both). Videos are generated by integrating the learned ODE from t=1 to t=0. The velocity network v_Θ is a DiT with an input projection g_1, N sequential transformer blocks, and an output projection g_2:

$$v_\Theta(x_t,t,c) = g_2 \circ B^{N-1} \circ \cdots \circ B^0 \circ g_1(x_t,t,c). \tag{2}$$

Each block B^i consists of multi-head self-attention (SA, H_{sa} heads) over the full spatiotemporal token sequence with 3D rotary embeddings [32], multi-head cross-attention (CA, H_{ca} heads) to text and image conditions, and a feed-forward network (FFN, hidden dimension d_{ffn}). Timestep conditioning enters via adaptive modulation [28].

3.2 Spatial-Temporal Aware Width Pruning

Attention heads in video DiTs play distinct roles: some process spatial information within temporal slices, while others capture temporal dynamics across slices. Standard magnitude-based pruning

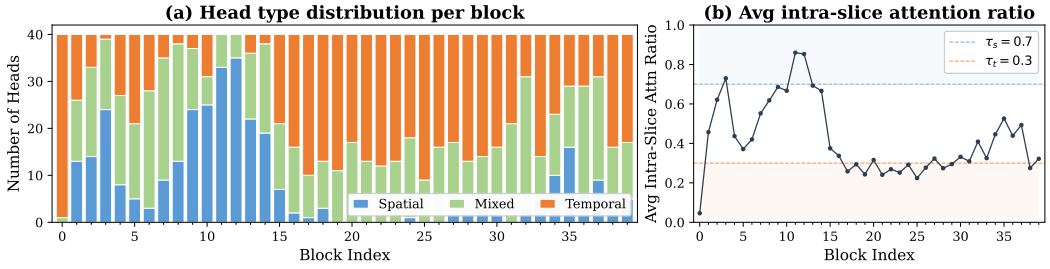

Figure 2: **Spatial-temporal head analysis of Wan2.1-14B. (a)** Distribution of head types across blocks, classified by the intra-slice attention ratio (Eq. (4)) with thresholds τ_s=0.7 and τ_t=0.3. **(b)** Average intra-slice attention ratio within each block. Shallow blocks (3–12) are dominated by spatial heads, whereas mid-to-deep blocks (15–31) are dominated by temporal heads. This variation motivates spatial-temporal aware importance scoring and temporal head protection.

overlooks these roles and risks disproportionately removing temporal heads, whose activations tend to be smaller despite their importance for motion coherence. We address this issue through spatial-temporal aware importance scoring and temporal head protection.

Attention Head Pruning. Each SA head h_j^i in block B^i contributes $\Delta y_j = W_{O,j}^i \cdot \tilde{\mathbf{v}}_j^i$ to the block output, where $\tilde{\mathbf{v}}_j^i$ denotes the attention-weighted value and $W_{O,j}^i$ is the corresponding slice of the output projection. Removing head j induces an output perturbation bounded by $\|W_{O,j}^i\|_F \cdot \|\tilde{\mathbf{v}}_j^i\|_F$. Motivated by the activation-weighted magnitude criterion in WANDA [34], we use this bound as an importance score that requires neither gradients nor leave-one-out evaluation. We aggregate importance scores over a calibration set $\mathcal{D}$, using $w(t) = \log(1 + t/T)$ to assign greater weight to high-noise timesteps:

$$s_{\text{sa}}^{i,j} = \frac{1}{|\mathcal{D}|} \sum_{(x,c)\in\mathcal{D}} \sum_{t=1}^{T} w(t) \cdot \left\|\tilde{\mathbf{v}}_j^i(x_t, c)\right\|_F \cdot \left\|W_{O,j}^i\right\|_F . \tag{3}$$

Cross-attention importance $s_{\text{ca}}^{i,j}$ is computed analogously; for I2V, text and image contributions are summed ($s_{\text{ca}}^{i,j} = s_{\text{ca,text}}^{i,j} + s_{\text{ca,img}}^{i,j}$). A limitation of magnitude-based scoring is that temporal heads, whose attention spreads across all F_l slices, have lower per-position magnitudes and are systematically underscored.

After VAE compression and patch embedding, a video is represented by $F_l L$ tokens arranged into F_l temporal slices, each containing L spatial tokens. We classify each SA head h_j^i using its *intra-slice attention ratio*, which measures the fraction of attention assigned to tokens within the query's temporal slice:

$$r_{\text{intra}}^{i,j} = \frac{1}{|\mathcal{Q}|} \sum_{q\in\mathcal{Q}} \frac{\sum_{k\in\mathcal{S}(q)} A_{q,k}^{i,j}}{\sum_k A_{q,k}^{i,j}}, \quad \text{type}(h_j^i) = \begin{cases} \text{spatial}, & r > 0.7, \\ \text{temporal}, & r < 0.3, \\ \text{mixed}, & \text{else}, \end{cases} \tag{4}$$

where $A_{q,k}^{i,j}$ denotes the post-softmax attention weight from query token q to key token k in head j of block i, $\mathcal{S}(q)$ contains the tokens in the same temporal slice as q, and $\mathcal{Q}$ is a subset of query positions sampled uniformly across temporal slices. As shown in Figure 2, shallow blocks (3–12) are dominated by spatial heads, whereas mid-to-deep blocks (15–31) are dominated by temporal heads.

To correct the magnitude bias, we scale temporal head scores by α_{temp}>1:

$$\tilde{s}_{\text{sa}}^{i,j} = \begin{cases} \alpha_{\text{temp}} \cdot s_{\text{sa}}^{i,j}, & \text{if type}(h_j^i) = \text{temporal}, \\ s_{\text{sa}}^{i,j}, & \text{otherwise}. \end{cases} \tag{5}$$

Each block retains $K = \max(K_{\min}, \lceil H_{\text{sa}}(1-p) \rceil)$ heads ranked by $\tilde{s}^{i,j}$, where H_{sa} is the original head count, p is the target pruning ratio, and $K_{\min}$=8. All blocks retain the same number of heads, but their identities vary across blocks. Temporal protection preserves temporal heads in temporal-dominant blocks, while spatial heads remain favored in spatial-dominant blocks.

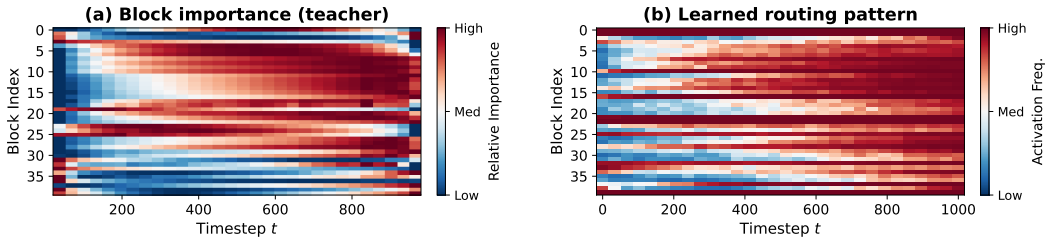

Figure 3: **From block importance to learned routing.** **(a)** Teacher block contributions across denoising timesteps, measured by the residual norm $\|B^i(x^{(i)}) - x^{(i)}\|_2$ and independently normalized to $[0, 1]$ within each block for visualization. Larger residual norms indicate greater changes to the hidden state. Early and boundary blocks contribute consistently, while middle blocks exhibit timestep-dependent variation. **(b)** Block activation frequencies under the learned router. Routing reflects the teacher's importance patterns by selectively skipping less important blocks. The budget allocates more blocks to high-noise steps for structure recovery and fewer to low-noise steps for detail refinement.

FFN Dimension Pruning. FFN neurons contribute additively to the output, but importance-based selection alone can retain functionally redundant neurons. We therefore combine importance scoring with diversity-aware selection. Attention heads require calibration to account for data-dependent attention patterns. For FFN neurons, we instead exploit the factored up/down projection structure to define a purely structural importance score:

$$s^{i,k}_{\text{ffn}} = \|W^i_{\text{up},k,:}\|_2 \cdot \|W^i_{\text{down},:,k}\|_2, \tag{6}$$

where $W^i_{\text{up},k,:}$ and $W^i_{\text{down},:,k}$ denote the k-th row of the up-projection and the k-th column of the down-projection, respectively. To avoid retaining neurons with nearly identical behavior, we define a signature $\mathbf{z}^i_k = [W^i_{\text{up},k,:}\ ;\ W^i_{\text{down},:,k}] \in \mathbb{R}^{2d}$. We then consider candidates in descending order of $s^{i,k}_{\text{ffn}}$ and skip any candidate whose signature has a cosine similarity greater than τ_{ffn} to that of an already selected neuron:

$$\text{skip neuron } k \ \text{ if } \ \max_{j \in \mathcal{K}} \frac{\mathbf{z}^i_k \cdot \mathbf{z}^i_j}{\|\mathbf{z}^i_k\| \|\mathbf{z}^i_j\|} > \tau_{\text{ffn}}, \tag{7}$$

where $\mathcal{K}$ is the set of neurons selected so far. If the number of selected neurons falls short of the target budget after one pass, τ_{ffn} is progressively relaxed and the skipped candidates are revisited until the budget is met. This yields a compact FFN where retained neurons are both individually important and functionally diverse. Dimensions are aligned to multiples of 128 for tensor-core efficiency. After width pruning, we construct the student v_Φ by extracting the retained weight submatrices from the teacher. The student retains all N blocks, with reduced attention and FFN widths.

3.3 Content-Adaptive Block Routing

Width pruning applies the same compressed architecture to every input and denoising step, leaving variations in denoising difficulty and visual content unexploited. High-noise steps establish global structure, whereas low-noise steps refine details; complex multi-object motion also relies on different blocks than a near-static landscape. Static block pruning [29] fixes the block set and does not account for either variation. We separate block selection into two decisions: a timestep-dependent *budget schedule* determines *how many* blocks to activate, and a timestep- and content-conditioned *router* determines *which* blocks to execute.

To determine how many blocks to activate, we use a linear budget schedule:

$$K(t) = K_{\min} + (K_{\max} - K_{\min}) \cdot \frac{t}{T}, \tag{8}$$

where $K_{\min}$ and $K_{\max}$ specify the minimum and maximum budgets. The schedule allocates more blocks to high-noise steps ($t \to T$) for recovering global structure, including scene layout, object placement, and motion trajectories, and fewer blocks to low-noise steps for local refinement. Budget values are provided in §4.1.

Given this budget, the router $\mathcal{R}_\phi$ determines which blocks to execute by mapping timestep and content embeddings to N block-wise importance logits:

$$\mathcal{R}_\phi(t, c) = W_3\, \sigma(W_2\, \sigma(W_1\, [\text{SinEmb}(t);\, e_c] + b_1) + b_2) + b_3, \tag{9}$$

where $\text{SinEmb}(t)$ denotes a sinusoidal timestep embedding, e_c is a content embedding, and σ is the SiLU activation. For I2V, $e_c = \text{CLIP}_{\text{cls}}(c_{\text{img}})$ summarizes the reference image; for T2V, e_c is derived from noisy latent statistics. Timestep conditioning captures which blocks are important at each noise level, while content conditioning enables *per-sample* adaptation. At the same timestep and budget, a fast-motion video may favor temporal-processing blocks, whereas a near-static scene may favor spatial-processing blocks. We convert the resulting logits $\mathbf{r} = \mathcal{R}_\phi(t, c) \in \mathbb{R}^N$ into a hard binary mask by selecting the $K = K(t)$ highest-scoring blocks:

$$m_i^{\text{hard}} = \begin{cases} 1, & \text{if } r_i \in \text{top-}K(\mathbf{r}), \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

To train the router through this discrete selection, we use the straight-through estimator (STE) [1] with the soft surrogate $m_i^{\text{soft}} = \text{sigmoid}(r_i)$:

$$m_i = m_i^{\text{hard}} + m_i^{\text{soft}} - \text{sg}(m_i^{\text{soft}}), \tag{11}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. The forward pass uses the hard binary mask, while the backward pass propagates gradients through the sigmoid surrogate, allowing the router to learn from the distillation loss. The resulting mask controls block execution through a residual blend:

$$x^{(i+1)} = m_i \cdot B^i(x^{(i)}) + (1 - m_i) \cdot x^{(i)}. \tag{12}$$

When $m_i = 0$, the block is skipped and $x^{(i+1)} = x^{(i)}$. At inference, skipped blocks are *not executed*, yielding real wall-clock savings. The first and last blocks (B^0, B^{N-1}) are always active, as they interface with the input and output projections.

3.4 Progressive Training Pipeline

Width pruning and block routing compress orthogonal dimensions, but optimizing them jointly from scratch creates a circular dependency: the router cannot learn meaningful block selection until the student produces reasonable features, yet the student cannot compensate for missing blocks until the router makes informed decisions. We break this dependency through progressive training, first recovering the width-pruned student and then jointly optimizing the student and router.

Stage I: Width Distillation. The width-pruned student v_Φ is first trained *without routing* via knowledge distillation from the frozen teacher v_Θ. The training objective combines four complementary loss terms:

$$\mathcal{L}_{\text{width}} = w_f \cdot \mathcal{L}_{\text{feat}} + w_t \cdot \mathcal{L}_{\text{TFM}} + w_d \cdot \mathcal{L}_{\text{DFM}} + w_\tau \cdot \mathcal{L}_{\text{temp}}. \tag{13}$$

The **feature matching loss** $\mathcal{L}_{\text{feat}}$ aligns teacher and student intermediate representations at randomly sampled blocks, using a linear projection P^i when dimensions differ:

$$\mathcal{L}_{\text{feat}} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left\| P^i(f_\Phi^i) - \text{sg}(f_\Theta^i) \right\|_2^2, \tag{14}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator. Outlier token positions beyond 3σ from the mean are masked to stabilize training. The **teacher velocity loss** $\mathcal{L}_{\text{TFM}} = \|v_\Phi - \text{sg}(v_\Theta)\|_2^2$ aligns student predictions with those of the teacher, while the **data velocity loss** $\mathcal{L}_{\text{DFM}} = \|v_\Phi - (\epsilon - x_0)\|_2^2$ additionally anchors the student to the ground-truth flow target, preventing the student from merely copying the teacher without learning the underlying data distribution. The **temporal consistency loss** [26, 9, 30] $\mathcal{L}_{\text{temp}} = \|\Delta_F(v_\Phi) - \text{sg}(\Delta_F(v_\Theta))\|_2^2$ matches inter-slice differences $\Delta_F(\cdot)$ between student and teacher to explicitly supervise motion coherence.

Stage II: Joint Width-Routing Training. Starting from the Stage I checkpoint, we introduce the router $\mathcal{R}_\phi$ and jointly optimize it with the student v_Φ. We retain the objective in Eq. (13), but restrict feature matching to *active* blocks ($m_i > 0.5$), excluding skipped blocks from intermediate supervision:

$$\mathcal{L}_{\text{joint}} = w_f \cdot \mathcal{L}_{\text{feat}}^{\text{active}} + w_t \cdot \mathcal{L}_{\text{TFM}} + w_d \cdot \mathcal{L}_{\text{DFM}} + w_\tau \cdot \mathcal{L}_{\text{temp}}. \tag{15}$$

Table 1: **Text-to-video generation results on VBench.** We report VBench dimension scores (%). **Bold**: best among compressed methods. Underline: second best. Highlighted : our full method.

Method	#Act.Params	Steps	SubCon	BgCon	Aesth.	Img.Q	Mot.Sm	Dyn.	Avg.
Un-accelerated Text-to-video methods									
VideoCrafter2 [2]	1.4B	50×2	98.16	**98.48**	66.34	68.46	98.42	17.48	74.56
LTX-Video [10]	13B	40	**98.42**	97.76	66.80	**69.97**	**99.07**	23.93	75.99
CogVideoX1.5 [51]	5B	50×2	92.66	94.95	56.24	62.81	96.96	**46.11**	74.95
Wan2.1-T2V [36]	1.3B	50×2	96.91	97.16	66.26	67.49	98.52	33.89	76.70
Wan2.1-T2V [36] (Teacher)	14B	50×2	96.99	97.49	**68.76**	68.20	98.59	36.20	**77.70**
Compression methods on Wan2.1-T2V-14B									
NeoDragon [16]	10.5B$_{(25\%\downarrow)}$	50×2	94.33	96.28	**59.50**	**69.00**	97.44	**31.54**	**74.68**
ICMD [42]	9.80B$_{(30\%\downarrow)}$	50×2	71.42	87.46	29.13	37.66	97.20	15.35	56.37
F3-Pruning [33]	10.5B$_{(25\%\downarrow)}$	50×2	**97.84**	**96.50**	57.92	57.83	**98.55**	10.37	69.84
Distillation methods on Wan2.1-T2V-14B									
DMD2 [5]	14B	4	95.38	96.16	68.00	**70.54**	98.36	**40.66**	**78.18**
rCM [64]	14B	4	96.92	97.20	**69.87**	69.30	98.66	32.54	77.42
AccVideo [58]	14B	5	97.13	**97.85**	69.32	62.29	**98.85**	30.06	75.92
CausVid [54]	14B	4	**97.34**	97.17	68.54	69.14	98.36	34.17	77.45
PARE (Width-only)	9.8B$_{(30\%\downarrow)}$	50×2	97.36	97.50	67.81	71.39	**98.79**	31.90	**77.46**
PARE (Routing-only)	9.5B$_{(32\%\downarrow)}$	50×2	96.55	97.21	66.30	70.02	97.76	**35.24**	77.18
PARE	6.7B$_{(52\%\downarrow)}$	50×2	97.56	97.37	66.39	**71.55**	98.76	30.98	77.10
PARE + Step Distill.	7.7B$_{(45\%\downarrow)}$	4	**98.75**	**98.33**	**67.92**	69.86	98.74	28.40	77.00

Stage III: Step Distillation. The per-step compressed model from Stage II is compatible with step distillation methods [53, 52]. Reducing sampling from approximately 50 steps to 4 provides an additional multiplicative speedup on top of the per-step compression achieved in Stages I and II.

Compression Analysis. We estimate the total speedup by combining width compression, block routing, and sampling-step reduction. After width pruning, block B^i retains a fraction ρ^i of its original FLOPs, and routing activates only $K(t)$ blocks at timestep t. Let C_B denote the average FLOPs per block in the unpruned model, $m_i(t) \in \{0, 1\}$ the routing mask defined in Eq. (10), and C_{embed} the fixed cost of the input and output projections. The per-step cost is $\mathcal{C}(t) = C_{\text{embed}} + \sum_i m_i(t) \cdot \rho^i \cdot C_B$, giving the following expected speedup:

$$\text{Speedup} \approx \frac{N}{\bar{\rho} \cdot \bar{K}} \cdot \frac{S_{\text{orig}}}{S_{\text{distill}}} \cdot G, \tag{16}$$

where $\bar{\rho}$ denotes the mean width retention, $\bar{K}$ the mean number of active blocks, and $S_{\text{orig}}/S_{\text{distill}}$ the sampling-step reduction factor. The CFG multiplier G equals 2 when guidance is removed and 1 otherwise. With $\bar{\rho} \approx 0.7$ and $\bar{K} \approx 27$, width pruning and routing yield approximately $2\times$ per-step speedup. Combining this reduction with four-step sampling ($S_{\text{orig}}/S_{\text{distill}} = 50/4$) and $G = 2$ gives a projected total speedup of approximately $50\times$.

4 Experiments

4.1 Experimental Setup

Implementation Details. We evaluate PARE on the I2V-14B and T2V-14B variants of Wan2.1 [36], each with 40 DiT blocks, d=5120, H_{sa}=H_{ca}=40, and d_{ffn}=13824. For both tasks, we prune SA, CA, and FFN widths by 30%, targeting approximately 30% overall parameter reduction. Importance scores are computed using 200 calibration samples across 10 timestep bins, with temporal head protection factor α_{temp}=1.5. Head selection within each block follows the spatial-temporal aware scoring in §3.2, and retained FFN dimensions are aligned to multiples of 128 for tensor-core efficiency. The router uses budget bounds $K_{\min}$=20 and $K_{\max}$=35.

We train on 30K video-text pairs with precomputed VAE latents and T5 and CLIP embeddings. Stage I (width distillation) and Stage II (joint width-routing training) each run for 7,000 steps on 4×H200 GPUs using FSDP, and a batch size of 4. The student and router learning rates are 3×10^{-5} and 5×10^{-4}, respectively, with cosine decay and a 100-step warmup. Loss weights are

Table 2: **Image-to-video generation results on VBench.** Same evaluation protocol as Table 1.

Method	#Act.Params	Steps	SubCon	BgCon	Aesth.	Img.Q	Mot.Sm	Dyn.	Avg.
Un-accelerated Image-to-video methods									
SEINE [3]	0.9B	250×2	95.02	95.75	59.21	**70.55**	96.65	25.63	73.80
I2VGen-XL [59]	1.4B	50×2	**96.65**	**97.27**	65.45	69.49	**98.43**	20.28	74.60
DynamiCrafter [48]	1.4B	50×2	95.39	95.37	**66.38**	69.09	97.46	31.97	75.94
CogVideoX1.5 [51]	5B	50×2	94.53	95.13	62.20	69.44	98.35	34.49	75.69
Step-Video-TI2V [13]	30B	50×2	95.36	96.12	63.74	68.38	97.40	37.34	76.39
Wan2.1-I2V [36] (Teacher)	16.2B	50×2	95.23	96.05	65.03	70.47	98.19	**42.54**	**77.92**
Compression methods on Wan2.1-I2V-14B									
NeoDragon [16]	12.2B$_{(25\%\downarrow)}$	50×2	**93.45**	**94.67**	**59.12**	**67.40**	97.02	30.23	**73.65**
ICMD [42]	11.3B$_{(30\%\downarrow)}$	50×2	72.00	86.65	30.72	45.45	**98.93**	12.98	57.79
F3-Pruning [33]	12.2B$_{(25\%\downarrow)}$	50×2	71.93	85.00	35.70	63.05	93.97	**55.67**	67.55
Distillation methods on Wan2.1-I2V-14B									
DMD2 [5]	16.2B	4	94.00	95.00	63.74	69.64	97.47	**38.59**	76.40
AccVideo [58]	16.2B	5	**95.62**	**96.02**	63.09	69.28	**97.88**	18.21	73.35
CausVid [54]	16.2B	4	94.32	95.03	**64.87**	**70.85**	97.35	38.12	**76.76**
PARE (Width-only)	11.3B$_{(30\%\downarrow)}$	50×2	94.86	95.25	63.13	**70.90**	98.39	**40.00**	**77.09**
PARE (Routing-only)	10.9B$_{(32\%\downarrow)}$	50×2	**96.50**	94.38	**65.07**	68.17	98.33	36.86	76.55
PARE	7.7B$_{(52\%\downarrow)}$	50×2	95.34	95.36	62.12	70.57	**98.56**	35.49	76.24
PARE + Step Distill.	8.9B$_{(45\%\downarrow)}$	4	95.60	**95.98**	62.97	68.86	98.32	33.84	75.93

$(w_f, w_t, w_d, w_\tau) = (10, 6, 1, 4)$ for I2V and $(10, 6, 1, 8)$ for T2V. For Stage III (step distillation), we use the Self-Forcing framework [14] and train for 1,200 steps with a learning rate of 2×10^{-6}.

Baselines. We compare against two categories of acceleration methods: (1) *structural compression*, including NeoDragon [16], ICMD [42], and F3-Pruning [33]; and (2) *step distillation*, including DMD2 [52], AccVideo [58], CausVid [54], and rCM [64], which reduce sampling steps while retaining the full model. We also report unaccelerated video generation models for reference. To isolate the contributions of width pruning and block routing, we include two variants: *Width-only*, the pruned student without routing, and *Routing-only*, the full-width model with block routing. The unpruned Wan2.1-14B [36] with 50-step sampling serves as the teacher upper bound.

Evaluation. We evaluate generation quality using VBench [15], reporting six dimensions: Aesthetic Quality, Imaging Quality, Motion Smoothness, Dynamic Degree, Background Consistency, and Subject Consistency, together with their normalized average. Inference uses the FlowUniPC scheduler [62] with 50 sampling steps and a classifier-free guidance scale of 5.0.

4.2 Quantitative Results

Text-to-Video. Table 1 reports results on Wan2.1-T2V-14B [36]. PARE achieves quality close to the teacher while substantially outperforming the structural compression baselines at similar or lower compression ratios. These include NeoDragon [16], which prunes along the depth dimension, and ICMD [42] and F3-Pruning [33], which lack spatial-temporal awareness. Combining PARE with step distillation maintains generation quality with only four sampling steps, confirming that per-step compression and step reduction provide complementary acceleration.

Image-to-Video. Table 2 shows a similar trend on I2V. PARE closely preserves teacher quality while halving per-step computation. It substantially outperforms NeoDragon [16] despite a greater reduction in active parameters, suggesting that depth-only pruning leaves substantial intra-block redundancy unexploited. Compared with full-model step distillation methods, PARE achieves quality competitive with DMD2 [52] and CausVid [54] at a fraction of their per-step computational cost. These results highlight the parameter efficiency of jointly compressing width and depth, rather than relying solely on sampling-step reduction.



Figure 4: **Qualitative comparison** between the teacher, PARE, and PARE + Step Distill.

4.3 Qualitative Results

Figure 4 compares representative I2V and T2V results. For I2V, the teddy bear sequence demands fine texture and coherent motion. PARE preserves the fur detail and climbing motion, although minor artifacts appear in later frames. Step distillation resolves these artifacts, suggesting that it provides additional refinement beyond reducing the sampling steps. For T2V, the aerial fantasy landscape requires coherent global structure. PARE preserves the composition and color fidelity of the teacher output, with quality maintained after distillation to four sampling steps.

4.4 Ablation Studies

We ablate the main design choices on I2V following §4.1, with additional T2V results for router conditioning. Highlighted rows denote the proposed configurations.

Table 3: Width pruning at a 30% pruning ratio.

Strategy	Img.Q↑	Aesth.↑	BgCon↑	Avg.↑
Uniform	60.04	57.22	92.30	71.78
PARE	70.90	63.13	95.25	77.09

Table 4: Routing versus random block pruning.

Strategy	Img.Q↑	Aesth.↑	BgCon↑	Avg.↑
Random	66.98	58.43	93.63	73.08
PARE	68.17	65.07	94.38	76.55

Width pruning strategy. Table 3 shows that spatial-temporal aware selection preserves quality better than uniform width pruning. A fixed pruning ratio does not account for the distinct functions of attention heads and FFN neurons. Temporal head protection counteracts the magnitude bias toward spatial heads, reducing the risk of discarding cross-frame interactions. Meanwhile, importance-plus-diversity selection avoids allocating FFN capacity to redundant neurons. Together, these mechanisms help retain complementary spatial and temporal processing within the compressed model.

Learned block selection. Learned routing outperforms random block pruning in Table 4 because blocks are not interchangeable across the denoising trajectory. Random removal can discard transformations that are important at a particular noise level, while retaining less useful ones. The router learns block scores through the distillation objective, favoring selections that preserve the teacher's predictions. This interpretation is consistent with Figure 3, where block contributions vary across timesteps, and explains why selecting which blocks to execute matters beyond reducing depth alone.

Table 5: Router conditioning for I2V and T2V tasks.

Router input	Img.Q↑	Aesth.↑	BgCon↑	Avg.↑
I2V-$\mathcal{R}_\phi(t)$	69.71	61.33	95.42	75.40
I2V-$\mathcal{R}_\phi(t, c_{\text{img}})$	70.57	62.12	95.36	76.24
T2V-$\mathcal{R}_\phi(t)$	71.27	65.80	97.91	76.84
T2V-$\mathcal{R}_\phi(t, c_{\text{lat}})$	71.55	66.39	97.37	77.10

Table 6: Stage II loss ablations on I2V, with both teacher and data velocity losses always kept active.

$\mathcal{L}_{\text{feat}}$	$\mathcal{L}_{\text{TFM}}$	$\mathcal{L}_{\text{DFM}}$	$\mathcal{L}_{\text{temp}}$	Img.Q↑	Mot.Sm↑	Avg.↑
✗	✓	✓	✗	65.61	96.47	72.10
✓	✓	✓	✗	70.65	98.19	75.34
✓	✓	✓	✓	70.57	98.56	76.24

Content conditioning. Table 5 shows that visual conditioning benefits both tasks, with a larger gain on I2V. Timestep information describes the noise level but cannot distinguish inputs with different visual content; it therefore assigns the same block scores to every sample at a given step. Content conditioning allows block selection to reflect sample-specific processing needs. The larger I2V benefit plausibly comes from the reference image's stable appearance and layout cues, whereas noisy latent statistics provide a less direct content signal for T2V.

Distillation objectives. Table 6 shows complementary benefits from feature matching and temporal consistency on top of the two velocity losses. Velocity supervision constrains the final prediction but leaves intermediate representations less directly guided. Feature matching supplies this guidance at active blocks as the student and router adapt jointly, helping the compressed network retain the teacher's internal representations. The temporal term adds a different constraint: matching inter-slice differences explicitly couples predictions across frames. This helps explain its additional benefit beyond matching features and output velocities.

4.5 Efficiency Analysis

We measure batch-one inference in BF16 on one H200 for 81-frame videos at 480×832 resolution. The denoising interval includes routing, top-K selection, dynamic dispatch, CFG, and scheduler updates, but excludes conditioning and decoding. Table 7 shows that width pruning and block routing provide complementary wall-clock gains: narrower blocks reduce the cost of each executed transformation, while routing reduces how many transformations are evaluated. Their combination achieves $1.78\times$/$1.68\times$ speedup on I2V/T2V, including routing overhead. The gap from the nominal $2\times$ compute estimate reflects the deployed schedule, which averages 32 rather than 27 active blocks after timestep shifting and boundary protection.

Table 7: Measured denoising time (s) and speedup on one H200, relative to the 50-step teacher for I2V and T2V.

	I2V		T2V	
Model	Time↓	Speedup↑	Time↓	Speedup↑
(a) 50 sampling steps				
Teacher	431.15	1.00×	435.14	1.00×
Width-only	295.23	1.46×	313.10	1.39×
Routing-only	344.33	1.25×	359.78	1.21×
PARE	242.53	1.78×	258.53	1.68×
(b) 4 sampling steps				
DMD2 (eager)	17.20	25.07×	18.36	23.70×
DMD2 (compiled)	15.78	27.32×	16.38	26.57×
PARE + step distill.	10.81	39.88×	13.08	33.27×

Width pruning also reduces resident DiT memory by approximately 30%; routing leaves candidate blocks loaded and therefore adds computational savings without further shrinking resident weights. With four-step distillation, eager dynamic PARE achieves $1.59\times$/$1.40\times$ denoising speedup over eager full-width DMD2 on I2V/T2V, and remains faster than its compiled counterpart. These comparisons isolate the benefit of per-step compression after sampling-step reduction. End-to-end gains are smaller because conditioning and decoding remain outside the compressed denoiser. Appendix A provides the full memory, compilation, end-to-end, and batching measurements.

5 Conclusion

We presented PARE, a framework that compresses video DiTs along both width and depth. The two axes demand fundamentally different strategies. Width pruning benefits from understanding the spatial-temporal roles of attention heads and the factored structure of FFN layers, whereas depth reduction benefits from adapting to both the denoising timestep and the visual content. PARE combines spatial-temporal aware width pruning with a learned content-adaptive block router, trained through a progressive pipeline that decouples the two objectives and avoids circular optimization difficulties. On Wan2.1-14B, PARE halves per-step computation for both image-to-video and text-to-video generation while closely preserving quality, and composes naturally with step distillation for acceleration along all three cost axes of video diffusion models.

Limitations. Our pruning ratios and router budgets are currently set as global hyperparameters and validated on Wan2.1-14B. Transferring them to architectures with substantially different block designs (e.g., MMDiT, U-ViT) may require re-calibration. The spatial-temporal head classification relies on attention patterns observed during calibration, which could shift under distribution changes such as significantly longer videos or unseen motion categories. PARE also exhibits lower Dynamic Degree scores despite largely preserving Motion Smoothness, suggesting reduced motion amplitude or frequency under compression. Temporal head protection does not fully eliminate this tendency toward lower-motion outputs. Additionally, while the progressive training pipeline is more efficient than training from scratch, it still requires access to a moderate-scale dataset and multi-GPU resources, which may limit adoption in resource-constrained settings.

6 Acknowledgments and Disclosure of Funding

This work was supported in part by the Australian Research Council under Projects DP240101848 and FT230100549. The work was supported in part by the National Key R&D Program of China (No. 2025YFE0201300).

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7310–7320, 2024.

[3] Xinyuan Chen, Yaohui Wang, Lingjun Zhang, Shaobin Zhuang, Xin Ma, Jiashuo Yu, Yali Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Seine: Short-to-long video diffusion model for generative transition and prediction. In *The Twelfth International Conference on Learning Representations*, 2023.

[4] Yuzhe Chen, Jiale Cao, Xuyang Liu, Jin Xie, Aiping Yang, and Yanwei Pang. Stdec: Spatio-temporal stability guided decoding for dllms. *arXiv preprint arXiv:2604.06330*, 2026.

[5] LightX2V Contributors. Lightx2v: Light video generation inference framework. `https://github.com/ModelTC/lightx2v`, 2025.

[6] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

[7] Xingtong Ge, Yutong Wang, Lunjie Zhu, Haitao Lin, Fangyu Lin, Yushi Huang, Xin Zhang, Yi Zhang, Yu Liu, and Jun Zhang. Salt++: Context-aligned post-training for few-step streaming multimodal generation. *arXiv preprint arXiv:2609.36995*, 2026.

[8] Xingtong Ge, Yi Zhang, Yushi Huang, Dailan He, Xiahong Wang, Bingqi Ma, Guanglu Song, Yu Liu, and Jun Zhang. Salt: Self-consistent distribution matching with cache-aware training for fast video generation. *arXiv preprint arXiv:2604.03118*, 2026.

[9] Amirhossein Habibian, Haitam Ben Yahia, Davide Abati, Efstratios Gavves, and Fatih Porikli. Delta distillation for efficient video processing. In *European Conference on Computer Vision*, pages 213–229. Springer, 2022.

[10] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[11] Jingxuan He, Busheng Su, and Finn Wong. Posegen: In-context lora finetuning for pose-controllable long human video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4780–4789, 2026.

[12] Chengjie Hong, Feixiang He, Yiheng Zeng, Lulu Kang, and He Wang. Safe-svd: Sensitivity-aware fidelity-enforcing svd for physics foundation models. *arXiv preprint arXiv:2605.17985*, 2026.

[13] Haoyang Huang, Guoqing Ma, Nan Duan, Xing Chen, Changyi Wan, Ranchen Ming, Tianyu Wang, Bo Wang, Zhiying Lu, Aojie Li, et al. Step-video-ti2v technical report: A state-of-the-art text-driven image-to-video generation model. *arXiv preprint arXiv:2503.11251*, 2025.

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

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

[16] Animesh Karnewar, Denis Korzhenkov, Ioannis Lelekas, Adil Karjauv, Noor Fathima, Hanwen Xiong, Vancheeswaran Vaidyanathan, Will Zeng, Rafael Esteves, Tushar Singhal, et al. Neodragon: Mobile video generation using diffusion transformer. *arXiv preprint arXiv:2511.06055*, 2025.

[17] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[18] Boyi Li, Yifan Shen, Yuanzhe Liu, Yifan Xu, Jiateng Liu, Xinzhuo Li, Zhengyuan Li, Jingyuan Zhu, Yunhan Zhong, Fangzhou Lan, et al. Toward cognitive supersensing in multimodal large language model. *arXiv preprint arXiv:2602.01541*, 2026.

[19] Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, et al. Sol-attn: Accelerating video generation inference via on-the-fly attention sparsification. *arXiv preprint arXiv:2607.24027*, 2026.

[20] Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024.

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

[22] Enhuai Liu, Yunke Wang, Yutong Wang, Changming Sun, and Chang Xu. Losa: Near-lossless sparse attention for training-free video diffusion acceleration. *arXiv preprint arXiv:2608.12032*, 2026.

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

[24] Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, et al. Palm: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. *arXiv preprint arXiv:2601.07060*, 2026.

[25] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[26] Zhengyao Lv, Chenyang Si, Tianlin Pan, Zhaoxi Chen, Kwan-Yee K Wong, Yu Qiao, and Ziwei Liu. Dual-expert consistency model for efficient and high-quality video generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14983–14993. IEEE, 2025.

[27] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15762–15772, 2024.

[28] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[29] Shitong Shao, Yufei Gu, and Zeke Xie. Fastlightgen: Fast and light video generation with fewer steps and parameters. *arXiv preprint arXiv:2603.01685*, 2026.

[30] Yifan Shen, Jian Xu, Boyi Li, Yuner Zhang, Tianjiao Yu, Bingxuan Li, Houze Yang, Rushi Wang, and Xu Cao. Chronovision: Temporal reasoning via latent state reconstruction. *arXiv preprint arXiv:2608.05631*, 2026.

[31] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

[32] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[33] Sitong Su, Jianzhi Liu, Lianli Gao, and Jingkuan Song. F^3-pruning: a training-free and generalized pruning strategy towards faster and finer text-to-video synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4961–4969, 2024.

[34] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[35] Tencent Hunyuan Foundation Model Team. Hunyuanvideo 1.5 technical report, 2025.

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

[37] Yutong Wang, Xingtong Ge, Enhuai Liu, Yunke Wang, Tianfan Xue, Xinyuan Chen, and Chang Xu. Pulsequant: Propagation-guided subspace correction for 4-bit video diffusion transformers. *arXiv preprint arXiv:2609.33384*, 2026.

[38] Yutong Wang, Xingtong Ge, Enhuai Liu, Yunke Wang, Tianfan Xue, Yu Qiao, Yaohui Wang, Xinyuan Chen, and Chang Xu. Vdot++: Unified few-step video generation via unbalanced optimal transport distillation. *arXiv preprint arXiv:2610.03221*, 2026.

[39] Yutong Wang, Haiyu Zhang, Tianfan Xue, Yu Qiao, Yaohui Wang, Chang Xu, and Xinyuan Chen. Vdot: Efficient unified video creation via optimal transport distillation. *arXiv preprint arXiv:2512.06802*, 2025.

[40] Zhuohan Wang, Haoran Ma, Tianyu Wu, Yuanlin Duan, Zichun Liao, and Jieming Yu. Solving every step is not enough: Milestone oracles reveal a composition gap in llm math reasoning. *arXiv preprint arXiv:2609.32235*, 2026.

[41] Tianyu Wu, Yu Yao, Zhenting Qi, Han Zheng, Zhuohan Wang, Haoran Ma, Lawrence Liao, Himabindu Lakkaraju, Ju Li, and Yilun Du. D-pace: Dynamic position-aware cross-entropy for parallel speculative drafting. *arXiv preprint arXiv:2605.18810*, 2026.

[42] Yiming Wu, Zhenghao Chen, Huan Wang, and Dong Xu. Individual content and motion dynamics preserved pruning for video diffusion models. *arXiv preprint arXiv:2411.18375*, 2024.

[43] Yushu Wu, Yanyu Li, Anil Kag, Ivan Skorokhodov, Willi Menapace, Ke Ma, Arpit Sahni, Ju Hu, Aliaksandr Siarohin, Dhritiman Sagar, Yanzhi Wang, and Sergey Tulyakov. Taming diffusion transformer for efficient mobile video generation in seconds. *arXiv preprint arXiv:2507.13343*, 2025.

[44] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

[45] Xi Xiao, Xingjian Li, Yunbei Zhang, Cheng Han, Tianming Liu, Tianyang Wang, Runmin Jiang, Jihun Hamm, Xiao Wang, and Min Xu. Layer-specific prompt fusion discovery via differentiable search in vision foundation models. In *European Conference on Computer Vision*, pages 391–409. Springer, 2026.

[46] Xi Xiao, Chenrui Ma, Yunbei Zhang, Chen Liu, Zhuxuanzi Wang, Yanshu Li, Lin Zhao, Guosheng Hu, Tianyang Wang, and Hao Xu. Not all directions matter: Towards structured and task-aware low-rank model adaptation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2132–2154, 2026.

[47] Xi Xiao, Yunbei Zhang, Xingjian Li, Tianyang Wang, Xiao Wang, Yuxiang Wei, Jihun Hamm, and Min Xu. Visual instance-aware prompt tuning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 2880–2889, 2025.

[48] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In *European Conference on Computer Vision*, pages 399–417. Springer, 2024.

[49] Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025.

[50] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *European conference on computer vision*, pages 289–304. Springer, 2018.

[51] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

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

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

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

[55] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Yuanzhe Liu, and Ismini Lourentzou. Core3d: Collaborative reasoning as a foundation for 3d intelligence. *arXiv preprint arXiv:2512.12768*, 2025.

[56] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Ying Shen, Kiet A Nguyen, Adheesh Sunil Juvekar, and Ismini Lourentzou. Silsa: Sliding-window slice latents for topology-preserving high-resolution 3d generation. *arXiv preprint arXiv:2610.02201*, 2026.

[57] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Onkar Susladkar, Yuanzhe Liu, Xiaona Zhou, and Ismini Lourentzou. Elsa3d: Elastic semantic anchoring for unified 3d understanding and generation. *arXiv preprint arXiv:2607.06565*, 2026.

[58] Haiyu Zhang, Xinyuan Chen, Yaohui Wang, Xihui Liu, Yunhong Wang, and Yu Qiao. Accvideo: Accelerating video diffusion model with synthetic dataset. *arXiv preprint arXiv:2503.19462*, 2025.

[59] Shiwei Zhang, Jiayu Wang, Yingya Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qin, Xiang Wang, Deli Zhao, and Jingren Zhou. I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv preprint arXiv:2311.04145*, 2023.

[60] Tianchen Zhao, Tongcheng Fang, Enshu Liu, Wan Rui, Widyadewi Soedarmadji, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation, 2024.

[61] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. *arXiv preprint arXiv:2410.03456*, 2024.

[62] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:49842–49869, 2023.

[63] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024.

[64] Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. *arXiv preprint arXiv:2510.08431*, 2025.

A Practical Inference Efficiency

This appendix supplements the quality evaluation with measured latency, memory, compilation, and batching results. We distinguish denoising-loop latency from end-to-end generation time and report separate benchmark configurations separately.

A.1 Measurement protocol and fifty-step inference

Unless stated otherwise, the measurements in this section use a single NVIDIA H200, BF16 precision, batch size one, and 81-frame videos at 480×832 resolution. The synchronized denoising interval includes router inference, top-K selection, dynamic Python block dispatch, classifier-free guidance (CFG), and scheduler updates. It excludes conditioning and video decoding. The teacher, width-only, routing-only, and joint models are evaluated with 50 denoising steps.

Table 8: Fifty-step denoising latency and throughput on Wan2.1. Speedup is relative to the teacher within each task. Throughput is computed as 3600 divided by the reported denoising time; it excludes the rest of the generation pipeline.

Task	Model	Time (s)↓	Speedup↑	Videos/h↑
I2V	Teacher	431.15	1.00×	8.35
	Width-only	295.23	1.46×	12.19
	Routing-only	344.33	1.25×	10.46
	PARE	242.53	1.78×	14.84
T2V	Teacher	435.14	1.00×	8.27
	Width-only	313.10	1.39×	11.50
	Routing-only	359.78	1.21×	10.01
	PARE	258.53	1.68×	13.92

Table 8 isolates the benefit of per-step compression from sampling-step reduction. Routing provides a further $1.22\times$ speedup on I2V and $1.21\times$ on T2V over the corresponding width-pruned backbone. The combined gains are smaller than a nominal $2\times$ FLOP estimate because realized execution depends on the sampled timesteps and protected blocks, as discussed in §B.2. These timings include dynamic dispatch rather than assuming that routing is free.

A.2 Memory and GPU utilization

Width pruning reduces the resident DiT allocation by physically shrinking its weight tensors. Routing skips execution but keeps the candidate blocks loaded, so it does not provide an additional reduction in resident weight memory. Table 9 separates DiT-resident allocation from the denoising peak; neither quantity should be equated with the whole-process memory reported for the caching experiments in §D.

Table 9: Memory and mean GPU utilization for the fifty-step benchmark. Memory is in GiB.

Task	Model	DiT resident	Denoising peak	GPU util. (%)
I2V	Teacher	31.49	36.70	99.98
	Width-only	22.11	26.14	99.92
	Routing-only	31.49	36.71	99.94
	PARE	22.11	26.16	99.84
T2V	Teacher	27.13	32.50	99.93
	Width-only	19.21	24.37	99.88
	Routing-only	27.13	32.51	99.86
	PARE	19.21	24.37	99.75

The similar width-only and joint-model memory footprints reflect this distinction: width determines the stored parameter volume, whereas routing reduces the executed work. High GPU utilization is maintained across these configurations, but utilization alone is not a measure of useful throughput or kernel efficiency.

A.3 Dynamic routing and compilation

Input-dependent block selection produces multiple execution paths, making a single static whole-model graph less straightforward to reuse. This does not make the compressed operators inherently incompatible with compilation. We evaluate a compiled fixed-route variant and eager execution with fully dynamic routing in a separate T2V benchmark (Table 10). The fixed route executes 29 width-pruned blocks and serves as a compilation diagnostic, not as a replacement for the content-adaptive model.

Table 10: Compilation diagnostic on T2V. Forward times refer to a conditional DiT forward. The first compiled call includes lazy compilation; the final column measures the 50-step denoising loop in the indicated execution mode.

	Conditional forward (s)			Denoising
Transformer path	Eager	Compiled	First call	(s)
Full-width Wan2.1	4.287	3.930	36.85	443.2
PARE, fixed 29-block route	2.257	2.021	26.91	245.8
PARE, dynamic eager	2.485	—	—	248.5

Compilation accelerates the fixed-route conditional forward by $1.12\times$. Relative to compiled full-width inference, fixed-route PARE achieves $1.94\times$ faster conditional forwards and $1.80\times$ faster denoising. Fully dynamic eager PARE remains $1.78\times$ faster in denoising than the compiled full-width baseline. Thus, the measured compression benefit persists even when the baseline benefits from compilation. The timings in this diagnostic are reported independently of Table 8.

A.4 Composition with four-step distillation

For the supplementary timing comparison, the full-width four-step DMD2 implementation was adapted from the Self-Forcing framework [14]; PARE combines the compressed model with four-step distillation. We additionally benchmark the full-width checkpoint with `torch.compile`, separating the first invocation from steady execution. PARE uses its deployed eager, dynamic-routing mode.

Table 11: Four-step inference timings. End-to-end time includes text/image conditioning, denoising, VAE decoding, T5 loading and offloading, and MP4 writing. The compiled steady measurement uses the second sample; first-call timings include lazy compilation. End-to-end times were not reported for the eager full-width baseline.

Task	Four-step system	Denoising (s)	End-to-end (s)
I2V	Full-width DMD2, eager	17.20	—
	Full-width DMD2, compiled first call	52.41	74.64
	Full-width DMD2, compiled steady	15.78	31.83
	PARE + step distillation, eager	10.81	25.52
T2V	Full-width DMD2, eager	18.36	—
	Full-width DMD2, compiled first call	65.85	78.89
	Full-width DMD2, compiled steady	16.38	29.43
	PARE + step distillation, eager	13.08	25.56

The comparison in Table 11 separates the contribution of per-step compression from that of using four sampling steps. PARE is $1.59\times$/$1.40\times$ faster in denoising than the eager full-width four-step baseline on I2V/T2V, and $1.46\times$/$1.25\times$ faster than its compiled steady counterpart. Relative to the respective 50-step teachers in Table 8, denoising is accelerated by $39.88\times$/$33.27\times$. End-to-end gains are smaller because conditioning, decoding, and output operations remain outside the compressed denoiser.

A.5 Heterogeneous-batch dispatch

Different samples in a batch can select different block subsets. In the heterogeneous-batch dispatch benchmark, exact routing takes 4.51 s and 9.06 s at batch sizes two and four, respectively, compared

with 6.19 s and 12.41 s when every sample executes all 40 width-pruned blocks. This preserves an approximately $1.37\times$ routing gain at both batch sizes. However, latency grows almost linearly with batch size, so these measurements do not show an additional throughput gain from batching. They support small heterogeneous batches; efficient large-batch serving remains an open implementation issue. These dispatch timings are distinct from full denoising-loop measurements.

B Routing Behavior and Compression Budgets

B.1 Stable and selectively executed blocks

The 50-step routing traces contain both consistently active blocks and blocks whose execution varies across the trajectory (Table 12). This pattern differs from simply truncating the network: the router preserves a stable set of transformations while selectively using other blocks.

Table 12: Selected statistics from the default 50-step routing traces. The counts describe these traces rather than universal activation frequencies across all inputs.

Task	Blocks active at all 50 steps	Examples of selectively active blocks
T2V	15	Block 33: 9/50; block 14: 22/50
I2V	23	Block 7: 14/50; block 12: 22/50

An additional diagnostic over 20 I2V inputs finds 5–9 distinct masks at intermediate timesteps $t \in [200, 700]$, with 5–8 blocks varying across inputs. Masks are more similar at the lowest and highest noise levels, where budget constraints and score margins limit changes in selection. Thus, the observed content adaptation affects a bounded subset of blocks: timestep determines the coarse budget, while visual content adjusts which blocks execute within that budget.

Related adaptation methods also depart from uniform treatment of model components and inputs: StructLoRA [46] filters task-irrelevant update directions and coordinates layers, layer-specific prompt fusion [45] discovers different fusion operators across depth, and ViaPT [47] generates input-dependent prompts. These methods adapt parameter updates or prompt representations, whereas PARE adapts the executed block subset within a structurally compressed video generator.

B.2 Nominal compression versus realized execution

Width pruning removes capacity within every executed block, while routing removes entire residual transformations. Their errors can accumulate, explaining why the combined model can have slightly lower average quality than either single-axis variant at a milder compression setting. Distributing compression across both dimensions avoids placing the full reduction on a single axis. For the nominal retained fractions used in the main comparison, $0.70 \times 0.68 \approx 0.48$, corresponding to approximately 52% reduction in the simplified compute model.

The nominal estimate and the measured deployment schedule must be distinguished [40, 50]. The former assumes approximately 27 active blocks out of 40. In the deployed 50-step benchmark, the shifted UniPC timesteps and protected boundary blocks yield an average of approximately 32 active blocks. Equation (16) should therefore be interpreted as a budget-dependent compute estimate, while Tables 8 and 11 report measured latency.

C Diagnostics on Additional Video Backbones

We examine whether the motivating head-role heterogeneity and timestep-dependent block importance also occur in HunyuanVideo and HunyuanVideo-1.5 [35]. These are structural and intervention diagnostics on those backbones.

C.1 Spatial and temporal head roles

Applying the same intra-slice attention-ratio thresholds as Eq. (4) identifies spatial, mixed, and temporal heads on both architectures (Table 13). The exact proportions differ from Wan2.1, indicating that the head assignments should be calibrated for each backbone rather than transferred unchanged.

Table 13: Head-role counts on additional backbones under the same attention-ratio thresholds. The analyzed head totals are 60×24 and 54×16, respectively.

Backbone	Spatial	Mixed	Temporal	Total
HunyuanVideo	63	328	1,049	1,440
HunyuanVideo-1.5	17	141	706	864

C.2 Block importance along real trajectories

We perform leave-one-block-out interventions on four real 50-step denoising trajectories for each backbone. Table 14 reports Spearman correlations of the resulting block-importance rankings. Rankings are relatively consistent across prompts at the same denoising stage, but change substantially between early and late stages. The results support timestep-dependent block selection beyond Wan2.1, while leaving the quality and efficiency of a fully trained cross-backbone PARE system to future evaluation.

Table 14: Spearman correlations of block-importance rankings on additional backbones. Cross-prompt ranges compare rankings at the same stage; the last column compares early and late denoising.

Backbone	Cross-prompt, same stage	Early versus late
HunyuanVideo	0.910–0.946	0.209
HunyuanVideo-1.5	0.770–0.889	−0.188

D Comparison with Caching and Dynamic Image DiTs

D.1 DeepCache deployment comparison

We benchmark the DeepCache [27] paths on Wan2.1 and HunyuanVideo-1.5 using one H200 and BF16 precision. This HunyuanVideo inference uses 848×480 resolution, 121 frames, and CFG scale 6 for the 50-step full-inference and caching configurations. Settings are matched within each backbone; the PARE + step-distillation rows use four steps. Table 15 reports observed process memory, which includes allocations beyond the resident DiT weights and differs from the memory scope in Table 9.

Table 15: Deployment measurements for caching and four-step PARE. Memory denotes observed process memory in GiB.

Backbone/task	System	Denoising (s)	End-to-end (s)	Memory
HunyuanVideo-1.5	Full inference	442	471.47	≈ 44.3
	DeepCache	225	261.10	≈ 66.9
Wan2.1 T2V	Full inference	435.14	457.98	≈ 33.54
	DeepCache	242.41	263.15	≈ 111.39
	PARE + step distill.	13.06	22.89	≈ 25.97
Wan2.1 I2V	Full inference	431.15	459.91	≈ 38.59
	DeepCache	241.55	268.91	≈ 118.19
	PARE + step distill.	10.52	25.86	≈ 27.82

On HunyuanVideo-1.5, DeepCache provides $1.96\times$ denoising and $1.81\times$ end-to-end speedup, while increasing observed process memory from approximately 44.3 to 66.9 GiB. Across three prompts and two seeds, its mean end-to-end speedup is $1.830\times$. On Wan2.1, caching similarly reduces latency but requires substantially more process memory in the tested implementation.

The approaches make different deployment trade-offs. DeepCache avoids retraining by reusing features across adjacent denoising steps, whereas PARE incurs training cost to reduce computation within each step. Four-step sampling leaves fewer opportunities for cross-step reuse; the measurements demonstrate PARE's composition with step distillation, but do not measure DeepCache combined with four-step distillation. Table 15 is therefore a deployment comparison.

D.2 Relationship to DyDiT

DyDiT [61] and PARE both learn to allocate computation adaptively across denoising steps and inputs, but differ in selection granularity. DyDiT uses timestep-conditioned gates to select attention heads and MLP channel groups, together with content-dependent token routing that bypasses selected MLP updates. PARE instead combines permanently pruned widths with whole-block routing conditioned on timestep and content, adapting execution depth while retaining dense token processing within active blocks. Thus, DyDiT exploits within-layer and spatial redundancy, whereas PARE removes entire attention and FFN transformations when a block is skipped, at the cost of coarser selection. For video generation, PARE further distinguishes spatial and temporal head roles during pruning and supervises inter-slice differences during distillation, explicitly addressing temporal information preservation. The methods therefore share a conditional-computation principle but make different trade-offs between fine-grained adaptivity, compact parameterization, and video-specific temporal modeling.

E Motion Preservation

The Dynamic Degree degradation in Tables 1 and 2 is an important quality limitation. Teacher-to-PARE scores decrease from 36.20 to 30.98 on T2V and from 42.54 to 35.49 on I2V, while Motion Smoothness remains close to the teacher. This pattern is consistent with reduced motion amplitude or frequency rather than primarily increased flicker. A plausible explanation is that aggressive channel and block compression favors more stable, lower-motion trajectories. Temporal-head protection addresses the under-selection of temporal heads, but does not eliminate this motion under-generation.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The method section and experiments can verify the contributions in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: I wrote the limitations at the end of the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: The assumptions and proof are complete.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The information provided by the current paper can reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [No]

Justification: The code hasn't been sorted out yet, so I didn't submit the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The authors provide all the details in the experiment details section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The authors provide all the details in the experiment details section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: I wrote the impacts at the end of the conclusion

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The license and terms of use are properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The new assets are well documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [No]

    Justification: We don't include human subjects' experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [Yes]

    Justification: The human subjects' experiments are under approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The method doesn't involve LLM as a core component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.