VORTA: Efficient Video Diffusion via Routing Sparse Attention

Wenhao Sun **Rong-Cheng Tu**[§] **Yifu Ding** **Zhao Jin**
Jingyi Liao **Shunyu Liu** **Dacheng Tao**[§]

College of Computing and Data Science, Nanyang Technological University, Singapore

{wenhao006, rongcheng.tu, n2409547h, zhao.jin}@ntu.edu.sg
{jingyi012, shunyu.liu, dacheng.tao}@ntu.edu.sg

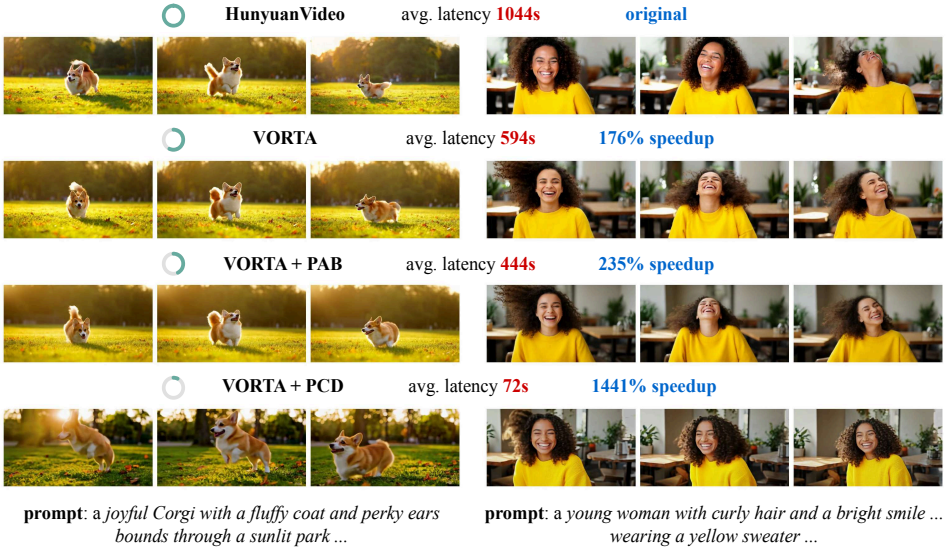


Figure 1: **VORTA** enables lossless acceleration of video diffusion transformers [16, 43], and remains compatible with other acceleration methods such as PAB [60] and PCD [44] for additional speedups.

Abstract

Video diffusion transformers have achieved remarkable progress in high-quality video generation, but remain computationally expensive due to the quadratic complexity of attention over high-dimensional video sequences. Recent acceleration methods enhance the efficiency by exploiting the local sparsity of attention scores; yet they often struggle with accelerating the long-range computation. To address this problem, we propose **VORTA**, an acceleration framework with two novel components: (1) a sparse attention mechanism that efficiently captures long-range dependencies, and (2) a routing strategy that adaptively replaces full 3D attention with specialized sparse attention variants. VORTA achieves an end-to-end speedup $1.76\times$ without loss of quality on VBench. Furthermore, it can seamlessly integrate with various other acceleration methods, such as model caching and step distillation, reaching up to speedup $14.41\times$ with negligible performance degradation. VORTA demonstrates its efficiency and enhances the practicality of video

§ Co-corresponding authors

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

diffusion transformers in real-world settings. Codes and weights are available at https://github.com/wenhao728/VORTA.

1 Introduction

Video Diffusion Transformers (VDiTs) have demonstrated impressive video generation performance, producing realistic and dynamic content [16, 33, 43, 51]. Despite this progress, VDiTs remain computationally expensive due to the inherently high-dimensional nature of video data, compounded by the quadratic complexity of attention operations. For example, recent HunyuanVideo [16] requires almost 1000 seconds (500 PFLOPS) to generate a 5-second 720p video at 24 frames per second (FPS) on an H100 GPU. To mitigate the high sampling cost, few-step sampling methods [39, 44, 52] leverage distillation on the self-consistency property of the probability flow ODE (PF-ODE) [38], achieving up to an $8\times$ reduction in sampling steps [56]. Other research [22, 26, 60] introduces intermediate feature caching to accelerate sampling without additional training.

Another promising direction investigates mitigating the quadratic computational complexity of the self-attention operation by leveraging its inherent sparse structure. The attention score distribution in VDiTs exhibits a dichotomy: local attentions and long-range attention. Local attentions [1, 53] tend to concentrate attention scores on a small subset of local keys around each query. Specifically, we observed that the spatially closest 4% of the keys (4,000 out of 108,000) contribute more than 80% of the total attention score as shown in Figure 2 (left). The remaining distant 96% keys, which dominate the computational cost, contribute less than 20%. Approaches [1, 50, 53, 55] to mitigate this inefficiency in attention with local behavior restrict interactions to nearby tokens and discard distant ones. The second regime, long-range attention, distributes interactions across the entire sequence to capture global context and long-range dependencies. In these attention heads, the closest keys account for less than 40% of the total attention score as shown in Figure 2 (right). Directly imposing local attention constraint on these inherently nonlocal heads for acceleration compromises the critical global context modeling necessary for effective video generation [51]. Many works [47, 55] resort to dense operation in nonlocal heads, which severely limits the overall acceleration gain. Alternative unstructured or semi-structured attention pruning methods [40, 46, 54] also struggle with quadratic-time sparsity detection or profiling, introducing substantial overhead during the forward pass. Therefore, effectively and efficiently accelerating the long-range attention components in VDiTs remains a critical, unresolved challenge.

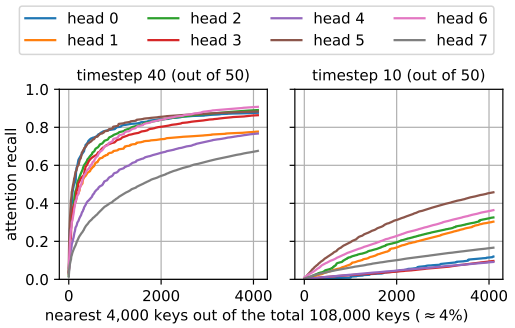


Figure 2: Attention scores recalled by the nearest keys. (*left*) Attention scores are predominantly concentrated within a local neighborhood. (*right*) The locality is less pronounced at earlier sampling steps. Results are from the 20th (of 60) layer in HunyuanVideo [16]. Only 8 out of the 24 attention heads are shown for clarity.

We observed that components exhibiting long-range dependencies are associated with high intra-sequence redundancy, whose tokens are highly similar. In this scenario, interactions with such redundant tokens are inefficient, as a few representative tokens can summarize their information. An example is provided in Figure 3 (middle), which depicts an intermediate generation result after the first few sampling steps. While high-level semantic structures, like spatial layouts and object motions, are primarily formed, the tokens have not acquired sufficient distinctiveness (*i.e.*highly redundant). This motivates our design of a token-level sparse attention

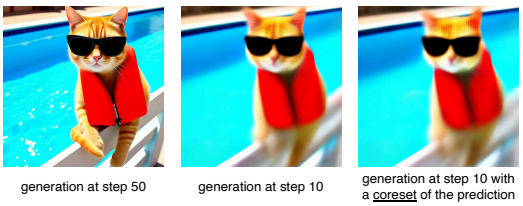


Figure 3: *(left)* Generation with the complete sampling process. *(middle)* Intermediate generation result. *(right)* Intermediate generation result with only core-set predictions.

to substitute the full-sequence computation when a compact core-set is sufficient to capture all necessary information. We propose a bucketed core-set selection (BCS) strategy to remove similar, redundant tokens and retain only the most representative ones for attention computation. As illustrated in Figure 3 (right), the output generated with core-set tokens retains semantic accuracy. Given that the core-set contains fewer tokens (*e.g.*, 50% of the full sequence), the attention cost is reduced quadratically (*e.g.*, to 25% of the original cost of the full sequence).

The implicit relationship between the VDiT's attention behaviors (*i.e.*, local or long-range dependencies) and the sampling steps, which correlates with the input signal strength, presents a challenge for the principled integration of sparse attention variants. Several approaches [47, 55] log attention score distributions through a reverse diffusion sampling process and reuse them. However, this is impractical in real-world settings where sampling configurations (*e.g.*, video size, sampling steps, or reverse diffusion solvers) frequently change, rendering the logged patterns obsolete and requiring re-tuning. To overcome this, we introduce a VORTA routing mechanism that dynamically selects the optimal sparse attention variant by leveraging the input signal-to-noise level. It provides precise control and adaptable support for diverse diffusion configurations, offering greater flexibility.

To conclude, our contributions are summarized as follows:

- We propose a novel core-set attention that effectively models long-range dependencies while theoretically reducing the attention cost by 75%.
- We introduce VORTA, an approach that integrates multiple sparse attentions into VDiTs for end-to-end acceleration without sacrificing performance. VORTA is compatible with various SDE/ODE solvers and network backbones.
- VORTA achieves a $1.76\times$ end-to-end speedup on VBench [11]. It is also compatible with other acceleration techniques, achieving overall speedups of up to $2.35\times$ with feature caching [60] and $14.41\times$ with consistency distillation [44].

2 Preliminary and related work

Flow matching and diffusion models. Flow Matching (FM) [21, 23] builds upon Continuous Normalizing Flows (CNFs) [5] and has since been integrated with the diffusion paradigm [10, 38], forming the foundation of many recent diffusion models [9, 12, 16, 32, 43, 61]. The core concept of FM involves a time-dependent velocity field (VF) v_t and a time-dependent diffeomorphism $\mathbf{x}_t \coloneqq t\mathbf{x}_1 + (1-t)\mathbf{x}_0$, known as a *flow*. The VF governs the evolution of the flow through an ordinary differential equation (ODE) $\mathrm{d}\mathbf{x}_t = v_t\left(\mathbf{x}_t\right)\mathrm{d}t$, where the flow $\mathbf{x}_t$ defines a probability density path p_t, starting from the simple prior $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and evolving towards the intractable target density p_1.

Lipman et al. [21] introduced Conditional Flow Matching (CFM) to optimize the neural network $v_t^\theta(\mathbf{x}_t)$ (*i.e.*VDiTs in this context) as follows:

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t, p_1(\mathbf{x}_1), p_0(\mathbf{x}_0)} \| v_t^\theta\left(\mathbf{x}_t\right) - \left(\mathbf{x}_1 - \mathbf{x}_0\right) \|^2. \tag{1}$$

Once optimized, it can start from $\mathbf{x}_0 \sim p_0$ and follow the ODE trajectory to generate samples.

3D self-attention of video diffusion transformers. Recent VDiTs [16, 18, 19, 33, 43, 51] employ 3D attention to capture spatio-temporal dependencies in video data. Given a video with F frames at a resolution of $H \times W$, the input is flattened into a sequence of length $L = F \times H \times W$, denoted as $\mathbf{H} \in \mathbb{R}^{L \times d}$, where each token is a d-dimensional vector. The self-attention is formulated as:

$$\mathrm{attn}(\mathbf{H}) = \mathrm{softmax}\left((\mathbf{H}\mathbf{W}_Q)(\mathbf{H}\mathbf{W}_K)^\top\right)(\mathbf{H}\mathbf{W}_V), \tag{2}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$ are the linear projection matrices. The attention operation is of complexity $\mathcal{O}(L^2 d) + \mathcal{O}(L d^2)$. In the context of high-resolution video processing, where the embedding dimension d is significantly smaller than the sequence length L, the overall complexity is dominated by the first term $\mathcal{O}(L^2 d)$, which scales unfavorably with sequence length. Despite employing techniques such as Variational Autoencoders (VAEs) [15, 34] and patchification [31], the sequence length L still reaches up to 100K for a 5-second 720p video in HunyuanVideo [16]. Consequently, attention operations dominate the computational cost, accounting for over 75% of the computation cost. Optimizing the attention computation is crucial for the efficiency of VDiTs.

Video diffusion acceleration. General diffusion acceleration methods include step distillation [24, 28, 35, 36, 39, 44], which reduces the number of sampling steps to as few as 4 to 8 with sufficient tuning, and feature caching [14, 22, 26, 58, 60], which avoids redundant computation by reusing intermediate features. In the case of VDiTs, the long sequence length leads to high attention costs. As a result, many studies have focused on optimizing attention for long sequences. Some recent approaches [47, 55] apply predefined sparse attention patterns to reduce computational overhead. Others [40, 46, 54] adopt online profiling to dynamically select attention patterns during inference. However, these sparse attention techniques depend on careful hyperparameter tuning for different configurations and/or introduce additional forward-pass complexity that offsets their speed benefits. VORTA addresses these challenges with a flexible design that is compatible with various diffusion configurations and model backbones, without incurring any additional inference-time overhead.

Acceleration by conditional computing. The concept of conditional computation has emerged as a powerful paradigm to accelerate neural networks by selectively activating only a subset of the parameters or operations for a given input. Bengio et al. [3] first introduced stochastic neurons in which parameter activation is conditional on their output. Subsequent methods selectively drop or gate layers [2, 27, 29, 37, 41, 45, 49], effectively utilizing a different, sparser network structure for each sample. In contrast to these prior efforts, our proposed VORTA integrates operation- and token-level sparsity and uses diffusion temporal dynamics to adaptively route computation.

3 VORTA: efficient video diffusion via routing sparse attention

This section introduces **VORTA** to accelerate VDiTs. We begin with a taxonomy of attentions in Section 3.1. Next, we present two core components: i) sparse attention variants that speed up specific attentions in Section 3.2; ii) a routing strategy that integrates these sparse attentions into pretrained VDiTs for end-to-end acceleration (Section 3.3).

3.1 Taxonomy of VDiT attentions

The attention sparsity has been briefly discussed in Section 1. We formally categorize attentions in VDiTs into three types:

- **Local attentions** focus on short-range interactions without attending to distant tokens. They are primarily responsible for fine-grained details during generation.
- **Long-range attentions** distribute their attention scores across the entire sequence. These attentions mainly capture high-level semantic information, including coarse layout and motion. Minor perturbations, such as merging similar tokens, are acceptable.
- **Pivotal attentions** maintain a global perceptual field while simultaneously refining local details. Small perturbations to these attentions can result in noticeable quality degradation.

All three types of attention coexist in VDiTs and are not mutually exclusive during the sampling: local attention may transition into nonlocal attention as the diffusion process evolves, and vice versa. We will provide supporting evidence for this taxonomy through experimental results in Figure 9.

3.2 Sparse attentions

Sliding window for local attentions. Sliding window attention [1, 53] proposes restricting each query token at position i to attend to its local neighborhood within the range $(i-w, i+w]$, where w is the window size. It provides an efficient alternative when the attention distribution is highly localized. However, its zigzag-shaped attention mask introduces computation bubbles in block-wise kernel implementations (*e.g.*, FlashAttention [6, 7]). This problem becomes more severe in three-dimensional video data, where the number of computation bubbles grows cubically. Zhang et al. [55] proposed sliding tile attention to tackle this problem. The 1D atten-

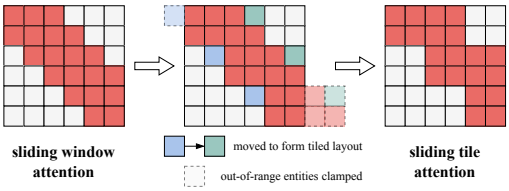


Figure 4: Illustration of converting a sliding window mask into a sliding tile mask. A 1D attention mask is shown for simplicity, with both the window size and tile size set to 2.

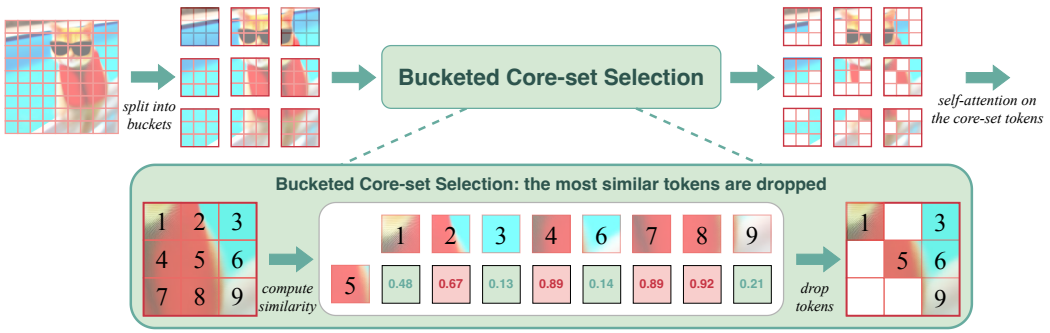


Figure 5: **Bucketed Core-set Selection (BCS).** For clarity, the 2D image is used for illustration; the actual inputs and buckets operate on 3D video data within the latent space. In this example, the top $k = 4$ tokens from each bucket are dropped.

tion mask is defined as: $\mathbf{M} = \{m_{i,j}\} = \{j \in (\tau(i) - w, \tau(i) + w]\}$, where $\tau(i) = \lfloor i/t \rfloor \cdot t + \lceil t/2 \rceil$ is the tile center, and t is the tile size. It can be interpreted as shifting the window of non-center queries to align with the query in the center of the tile, as illustrated in Figure 4. The attention mask for sliding tile attention is block-wise dense and offers greater hardware efficiency. Our implementation employs 3D sliding tile attention to model local interactions of quality attentions with detailed pseudo code in Appendix A.1.

Core-set selection for long-range attentions. It is observed that many attentions with large perceptual fields are less sensitive to perturbations [40], corresponding to the long-range attentions defined earlier. Based on this inherent token-level sparsity, we introduce core-set attention to drastically accelerate computation by actively pruning redundant tokens during the attention operation. A prototype core-set attention operation is defined as follows:

$$\text{coreset-attn}(\mathbf{H}) = \text{unpool} \circ \text{attn} \circ \text{pool}\,(\mathbf{H})\,, \tag{3}$$

where $\circ$ denotes the composition operator, meaning the connected operators are applied sequentially from right to left. $\text{pool}(\cdot)$ and $\text{unpool}(\cdot)$ operations compress a long sequence into a compact core-set and recover the original sequence length, respectively.

Direct application of standard average pooling often results in a significant degradation of generation quality (see Section 4.2), because its core assumption, that all tokens within the pooling kernel are sufficiently similar, is often violated. As an alternative, we implement the $\text{pool}(\cdot)$ operation using Bucketed Core-set Selection (BCS), as outlined in Figure 5. It introduces a novel, buketed approach to achieve token-level sparsity. BCS first divides the tokens into buckets. Intra-bucket similarity is computed exclusively between a designated center token (token '5' in this example) and its neighboring tokens. Tokens with the top-k highest similarity to the center are then pruned, and their representational information is merged into the center. Crucially, by eliminating inter-bucket similarity calculations, BCS achieves linear complexity $\mathcal{O}(L)$ for sequence length L, offering a significant computational advantage over $\mathcal{O}(L^2)$ sequence length pruning methods [4, 40] while maintaining competitive performance. A detailed complexity analysis is provided in Appendix A.1.

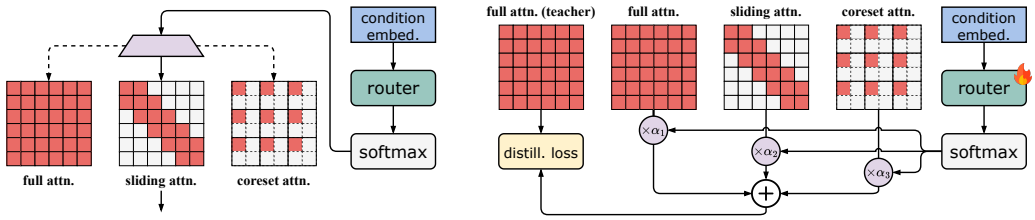
After applying the attention operation on the core-set tokens, we retrieve the original sequence length by scattering the center token back to the pruned tokens for the subsequent operations.

3.3 Signal-aware attention routing and detection

Attention router. Accurately identifying the optimal sparse attention variant is the remaining technical hurdle. We observe that attention behavior correlates strongly with the signal-to-noise ratio (SNR) of input features in Figure 2. To address this, we introduce a router implemented as a linear layer with diffusion timestep embedding $\mathbf{T} \in \mathbb{R}^d$ as input, following a softmax activation to output the gate values $\boldsymbol{\alpha}^{(n)}$ for dynamic attention variant selection:

$$\boldsymbol{\alpha}^{(n)} = \text{softmax}(\mathbf{T}\mathbf{W}_R^{(n)}), \tag{4}$$

where $\mathbf{W}_R^{(n)} \in \mathbb{R}^{d\times 3}$ denotes the linear projection matrix of the n-th transformer block. The gate values $\boldsymbol{\alpha}^{(n)} = \left[\alpha_1^{(n)}, \alpha_2^{(n)}, \alpha_3^{(n)}\right]$ quantify the suitability of full, sliding-window, and core-set



(a) routing between attention branches **(b) attention pattern detection**

Figure 6: **Overview of VORTA.** 1D masks are used to represent the attention branches for simplicity. (*a*) During inference, the router takes the condition embedding as input and selects the appropriate attention branch. In this example, it activates sliding attention. (*b*) During pattern detection, all attention branches are activated, and their outputs are aggregated using gate values from the router. The divergence between the full attention output and the routed output is used to update the router.

attention operations, respectively. This lightweight design adds just 0.1% to the model's parameters, while enabling the router to adaptively modulate gate values based on the current signal-to-noise ratio (SNR), without directly operating on the full set of diffusion tokens.

Inference-time routing strategy. The router activates the attention branch with the highest gate value, as illustrated in Figure 6 (a). It is formalized as a hard selection, defined as follows:

$$\mathbf{H}^{(n+1)} = \begin{cases} \text{sliding-attn}(\mathbf{H}^{(n)}) & \text{if } \alpha_2^{(n)} > \alpha_1^{(n)} \text{ and } \alpha_2^{(n)} > \alpha_3^{(n)} \\ \text{coreset-attn}(\mathbf{H}^{(n)}) & \text{if } \alpha_3^{(n)} > \alpha_1^{(n)} \text{ and } \alpha_3^{(n)} > \alpha_2^{(n)} \\ \text{attn}(\mathbf{H}^{(n)}) & \text{otherwise} \end{cases}, \tag{5}$$

where $\mathbf{H}^{(n)}$ represents the input features to the n-th transformer block. $\text{attn}(\cdot)$ is the standard full attention. $\text{sliding-attn}(\cdot)$ and $\text{coreset-attn}(\cdot)$ represent the sliding and core-set attentions, respectively, as defined in Section 3.2. Some operations, such as RoPE and FFN, have been omitted for clarity. Activating sparse attention branches yields a substantial speedup in attention computation. Although the full attention branch is activated in fewer than 0.2% of cases, it remains important for preserving model performance. We will empirically investigate these design choices in Section 4.2.

Router optimization. Inspired by recent advances in language modeling [49], we adopt a self-supervised optimization strategy, as illustrated in Figure 6 (b). In the forward process, the gate values are employed to weight the outputs of the three branches:

$$\mathbf{H}^{(n+1)} = \alpha_1^{(n)} \cdot \text{attn}(\mathbf{H}^{(n)}) + \alpha_2^{(n)} \cdot \text{sliding-attn}(\mathbf{H}^{(n)}) + \alpha_3^{(n)} \cdot \text{coreset-attn}(\mathbf{H}^{(n)}). \tag{6}$$

We introduce a distillation loss $\mathcal{L}_{\text{distill}}$, defined as the MSE between the routed output $\mathbf{H}^{(N)}$ and the original output $\mathbf{H}_{\text{org}}^{(N)}$ of the final block, to ensure that the routed outputs remain close to the original model outputs to preserve the pretrained performance. In addition, we adopt the conditional flow matching loss $\mathcal{L}_{\text{CFM}}$ from the VDiT pretraining stage as detailed in Section 2. To promote sparsity in routing decisions, we apply an L2 regularization term to gate values. The final loss function combines the conditional flow matching loss, the distillation loss, and the regularization term, weighted by hyperparameters λ_{distill} and λ_{reg}, respectively:

$$\mathcal{L} = \mathcal{L}_{\text{CFM}} + \lambda_{\text{distill}} \cdot \mathcal{L}_{\text{distill}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}, \tag{7}$$

$$\text{where } \mathcal{L}_{\text{distill}} = \text{MSE}\left(\mathbf{H}_{\text{org}}^{(N)}, \mathbf{H}^{(N)}\right) \text{ and } \mathcal{L}_{\text{reg}} = \sum_{n=1}^{N} \|\alpha_1^{(n)}\|^2. \tag{8}$$

Notably, the original parameters of the VDiTs are always frozen, and only the router is updated. We set $\lambda_{\text{distill}} = 20$ and $\lambda_{\text{reg}} = 0.02$ to balance the losses at a similar scale in our experiments. A detailed analysis of these hyperparameters is provided in Appendix B.1.

4 Experiments

Baselines. This section presents the evaluation of the text-to-video generation task. We evaluate VORTA with two recently open-sourced VDiTs: HunyuanVideo [16] and Wan 2.1 [43]. Our

Table 1: Quantitative comparison under standard baseline settings (bf16, 720p, 5s). C: caching; D: step distillation; S: sparse attention.

Models	Type	VBench ↑	Quality ↑	Semantic ↑	LPIPS ↓	Latency (s)	Speedup	Mem. (GB)
HunyuanVideo [16]	-	82.26	83.68	76.60	-	1043.85	1.00×	47.64
+ ARnR [40]	S	82.39	**83.85**	76.56	0.211	790.55	1.32×	78.15
+ STA [55]	S	82.33	83.56	77.39	0.201	676.39	1.54×	51.79
+ PAB [60]	C	82.40	83.80	76.81	0.186	815.51	1.28×	> 80
+ VORTA	S	**82.59**	83.74	77.95	**0.185**	594.23	1.76×	**51.15**
+ VORTA & PAB	S & C	82.56	83.60	**78.38**	0.195	**444.19**	**2.35**×	> 80
+ PCD [44]	D	81.17	82.56	75.35	**0.564**	125.98	8.29×	**47.64**
+ VORTA & PCD	S & D	**81.49**	**82.78**	**76.31**	0.575	**72.46**	**14.41**×	51.15
Wan 2.1 (14B) [43]	-	82.36	83.05	79.60	-	1304.82	1.00×	41.77
+ VORTA	S	**82.85**	**83.45**	**80.44**	0.222	**856.50**	**1.52**×	43.97

comparisons include sparse attention acceleration methods: STA [55], which adopts a predefined sparse attention pattern, and ARnR [40], which performs online profiling to determine the sparse attention pattern dynamically. We also compare VORTA against two orthogonal approaches: the caching-based method PAB [60] and the step distillation method PCD [44]. Additionally, we report results for VORTA combined with PAB and PCD, as these methods can be integrated. For the Wan 2.1 [43], we only compare the pretrained baseline and VORTA, since other methods have not yet released code to support this model.

Benchmarks and evaluation metrics. Following prior works [40, 60], we evaluate on the standard VBench prompt suite [11], which contains over 900 text prompts across 16 dimensions. The primary performance metric is the aggregated VBench score. We also report the VBench quality and semantic subscores to provide a more detailed breakdown. To assess the deviation of generation from the pretrained models, we use LPIPS [57] as a reference metric. For efficiency analysis, we measure video sampling latency of the VDiTs, excluding the time required for text encoding and VAE decoding. We report the relative speedup as $\Delta\text{latency}/(\text{latency}+1)$. Additionally, we record peak memory usage, which impacts practical deployment due to hardware constraints.

Implementation. We implement VORTA in PyTorch [30], using FlexAttention kernel for sliding attention and FlashAttention [6, 7] kernel for all other attention operations. Video samples are generated in 50 steps, except for PCD [44], which uses 6 steps. The videos are 5 seconds long at 720p resolution. Due to out-of-memory issues with PAB [60] at this resolution, we adopt sequential CPU offloading [42]. For fair comparison, the latency introduced by model loading and offloading is excluded in the experiment results. For router optimization, we use the Mixkit dataset [20], training for 100 steps with a learning rate of 10^{-2} and a batch size of 4. All experiments are conducted on H100 GPUs with 80GB of memory. Additional implementation details are provided in Appendix A.2.

4.1 Main results

Performance. Table 1 presents the quantitative evaluation results of the baseline methods and VORTA. The key observations are: i) All methods maintain high VBench scores, except PCD, which shows a 1-point drop due to aggressive compression of generation steps. ii) VORTA, with and without PAB, ranks first and second in VBench score and LPIPS on HunyuanVideo, respectively. iii) The good performance of VORTA on both the MMDiT-based [9] HunyuanVideo and DiT-based [31] Wan 2.1 demonstrates its generalizability across diffusion backbones. Figures 1 and 7 quantitatively depict the generations produced by ARnR [40], STA [55], VORTA, VORTA & PAB [60], and VORTA & PCD [44] on HunyuanVideo. All variants maintain high visual quality with vivid appearance and coherent motion, consistent with the quantitative metrics. More qualitative comparisons and complete VBench dimensional scores are provided in Appendix B.4 and Appendix B.5, respectively.

Efficiency. Among the sparse attention approaches, VORTA achieves the highest efficiency on HunyuanVideo, with a 1.76× speedup. For the other line of work, PAB achieves a 1.28× speedup on HunyuanVideo, but it requires over 1.7× additional memory, which makes it less practical for large models or high-resolution videos. In practice, running PAB on an 80GB GPU necessitates sequential

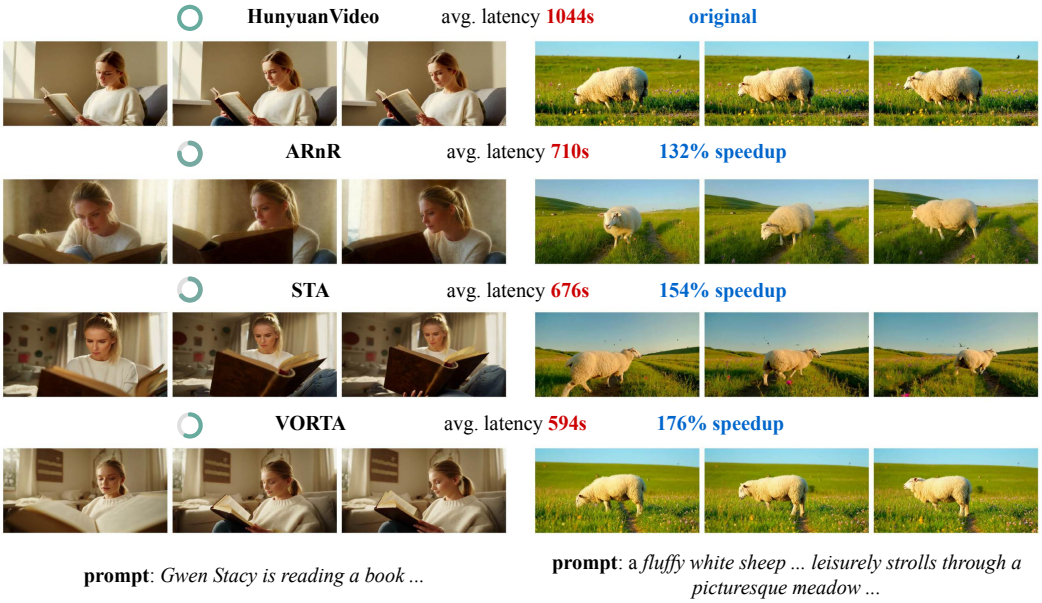


Figure 7: Qualitative comparison of sparse attention methods: ARnR [40], STA [55], and VORTA.

CPU offloading. The latency introduced by parameter loading and offloading nearly offsets the benefits of computation. The step distillation approach, PCD, achieves an $8.29\times$ speedup using only 6 sampling steps, with a mild drop in performance. While these methods offer different trade-offs in efficiency and performance, both can be effectively integrated with VORTA. When combined with PAB and PCD, VORTA achieves total speedups of $2.35\times$ and $14.41\times$, respectively.

Table 2: Quantitative comparison on Wan 2.1 [43] with different sampling configurations.

Models	Configuration	VBench				Latency (s)	Mem. (GB)
		Subject ↑	Consistency ↑	Aesthetic ↑	Imaging ↑		
Wan 2.1 (14B) [43]	UniPC (50 step) [59]	91.21	**75.49**	63.33	**63.99**	1359.76	41.77
+ VORTA		90.76	75.09	63.79	63.07	791.02	43.97
Wan 2.1 (14B) [43]	DPM++ (30 step) [25]	89.72	74.54	**64.43**	**63.99**	817.00	41.77
+ VORTA		**91.82**	75.13	64.32	63.95	**440.76**	43.97

Generalization to various schedulers. Unlike sparsity-inducing techniques that rely on offline profiling or hand-crafted heuristics to define a sparse execution strategy [40, 55], VORTA is designed to possess scheduler and step-count generalization, mirroring the flexibility of the baseline dense model. This obviates the need for costly re-profiling or hyperparameter re-tuning when changing sampling configurations. We demonstrate this capability by comparing the performance on the Wan 2.1 [43] using both the 50-step UniPC [59] and 30-step DPM++ [25] schedulers.

Table 2 presents the VBench dimensions [11] and efficiency metrics from these experiments, which were conducted on B200 GPUs. The consistent, lossless performance of VORTA relative to the dense model confirms its ability to generalize efficiently without added cost, an advantage enabled by its lightweight training, which heuristic-based methods like STA [55] and ARnR [40] do not achieve.

Runtime breakdown. Figure 8 further presents the average latency of individual components for the sparse attention methods. Here, "attn." denotes the isolated attention operation latency, whereas "attn. related" includes associated operations such as projections, RoPE, layer normalization, and other computations within the attention module.

Compared to ARnR [40], VORTA demonstrates superior efficiency. To preserve lossless generation, ARnR adopts a more conservative sparsity, resulting in higher latency in its "attn." component. Furthermore, its $\mathcal{O}(L^2)$ similarity computation introduces additional latency, as evidenced by the larger latency in its "attn. related" component. In contrast, VORTA adaptively identifies efficient sparse patterns and allows greater speedups. Its core-set attention also involves similarity computation; however, BCS has only $\mathcal{O}(L)$ complexity, incurring negligible additional cost.

VORTA also outperforms STA [55], which uses a predefined sliding attention pattern. The main bottleneck of STA occurs during the first 15 sampling steps (see bottom two bars), when it must expand the window size to capture long-range dependencies. Consequently, these steps do not benefit from attention sparsity. In contrast, VORTA adaptively routes attention branches to accommodate diverse interaction types while maintaining near-constant latency across all diffusion steps.

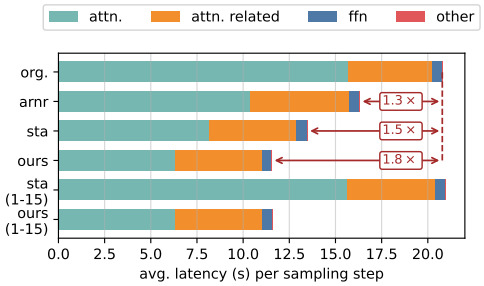


Figure 8: Runtime breakdown for sparse attention methods on HunyuanVideo [16].

Table 3: Evaluation of each VORTA component on Wan 2.1 (1.3B) [43].

Models	VBench ↑	Latency (s)	Speedup
Wan 2.1 (1.3B)	81.20	73.24	1.00×
w/o Sliding Attn.	80.25	65.14	1.12×
w/o Coreset Attn.	79.89	66.10	1.11×
w/o Full Attn.	77.14	59.34	1.23×
w/o Timestep Cond.	81.03	65.00	1.13×
w/ AP_{211}	77.08	**57.53**	**1.27**×
w/ AP_{121}	76.01	57.64	**1.27**×
w/ AP_{112}	75.94	57.55	**1.27**×
VORTA	**81.06**	58.42	1.25×

4.2 Ablation study

We conduct ablation studies on the Wan 2.1 (1.3B) [43] at its pretrained 480p resolution. All other experimental settings are identical to those used in the main experiment. At 480p, attention operation accounts for a smaller portion of the overall latency compared to 720p, resulting in less pronounced acceleration. Nevertheless, we adopt this smaller model and resolution to reduce the computational cost of benchmarking. The results are summarized in Table 3.

Attention branches. We evaluate the contribution of the attention branches individually by removing them. When the sliding attention branch is disabled, the router shifts more selections toward the full attention expert to preserve performance, resulting in an over 10% runtime increase, despite applying the same regularization level. Similarly, removing the core-set attention branch increases the latency by 14%. These results highlight the importance of both sparse branches in improving efficiency through the adaptive selection of attention patterns tailored to specific characteristics.

Given that the router selects the full attention branch in only 0.2% cases in VORTA (will be detailed in Section 4.3), we assess whether retaining it is necessary. As shown in Table 3, the removal of the full attention branch leads to a 4-point drop in the VBench score without any additional speedup. This suggests that the full attention branch remains crucial and validates the existence of the pivotal attention patterns defined in Section 3.1.

Timestep condition for signal-aware routing. We also examine the impact of including timestep information in attention routing. Removing the timestep embedding from the router input results in uniform attention branch selection across all diffusion steps, leading to slower 12%. In the absence of timestep conditioning, the router consistently selects the same attention expert. To maintain performance, the model predominantly routes to the full attention expert.

Bucketed core-set selection (BCS). We evaluate how much BCS improves performance compared to conventional average pooling. BCS sets the core-set size to 50% of the original sequence length. To ensure a fair comparison, we apply the same reduction ratio in the average pooling cases by configuring the pooling kernel sizes such that their product equals 2. To isolate the effect of compression along different dimensions, we evaluate three pooling kernel configurations: $(2, 1, 1)$, $(1, 2, 1)$, and $(1, 1, 2)$, denoted as AP_{211}, AP_{121}, and AP_{112}, respectively. Although average pooling offers slightly lower latency by being free from similarity computations, its performance drops significantly. When adjacent tokens differ significantly, naive merging causes information loss, often resulting in artifacts such as pixelation or blurring in the generation.

4.3 Attention patterns in VDiTs.

Figure 9 illustrates the routed attention branch assigned to each head, layer, and sampling step. It reveals a clear temporal pattern: earlier time steps tend to use more corset attention, while later time steps increasingly rely on sliding attention. Only a small fraction (about 0.2%) of attention heads are assigned to the full attention branch, highlighting the sparsity. As intuitively expected, earlier time steps likely focus on constructing high-level semantics rather than fine-grained details, whereas later time steps emphasize local interactions. Unlike auto-regressive or discriminative models [8, 13], VDiTs exhibit weaker layer-wise specialization; attention heads within the same layer are more evenly distributed across branches. A minor tendency is observed in later time steps where sliding attention is more frequently assigned to intermediate layers, while corset attention is more often assigned to the initial and final layers. One plausible explanation is that intermediate layers capture local details, while subsequent layers refine the overall representation quality. Step-wise visualizations and results for other models are provided in Appendix B.3.

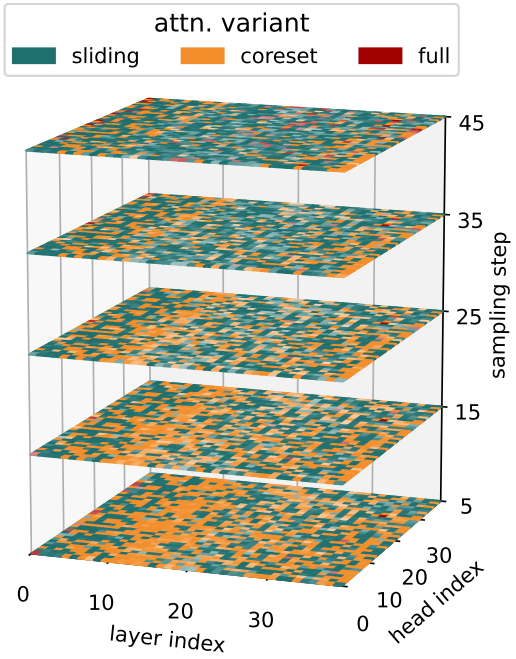


Figure 9: Optimized gate values for Wan 2.1 (14B) [43]. We visualize snapshots at sampling steps 5, 15, 25, 35, and 45 out of total 50 steps.

5 Conclusion

In this work, we presented VORTA, an efficient and generalized framework for accelerating diffusion transformers in video generation. VORTA reduces attention overhead by dynamically identifying attention patterns and routing them through appropriate sparse attention mechanisms. To further enhance efficiency, we introduced a bucketed coreset selection (BCS) strategy that improves the modeling of long-range dependencies. VORTA achieves a $1.76\times$ end-to-end speedup without compromising generation quality. Moreover, its high compatibility with existing acceleration techniques enables a combined speedup of up to $14.41\times$. We believe VORTA offers a practical and extensible solution, paving the way for broader adoption and future research in video generation.

Limitations. VORTA targets the attention mechanism, which accounts for over 75% of the total computation in high-resolution video generation. However, for tasks with short sequence lengths, such as image or low-resolution video generation, the attention overhead becomes less dominant. As a result, the potential acceleration achievable is limited. Additionally, this work focuses on the bidirectional generation paradigm. Other paradigms, such as autoregressive generation, are not directly supported and may require substantial adaptation. Additional discussion on failure cases, limitations, and border impacts is provided in Appendix C.

Acknowledgments and Disclosure of Funding

This project is supported by the National Research Foundation, Singapore, under its NRF Professorship Award No. NRF-P2024-001.

References

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

[2] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *CoRR*, abs/1511.06297, 2015.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[4] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *CVPR Workshops*. IEEE, 2023.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

[6] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024.

[7] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

[8] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024.

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*. PMLR, 2024.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[11] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models. In *CVPR*. IEEE, 2024.

[12] Haobo Jiang, Mathieu Salzmann, Zheng Dang, Jin Xie, and Jian Yang. SE(3) diffusion model-based point cloud registration for robust 6d object pose estimation. In *NeurIPS*, 2023.

[13] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *NeurIPS*, 2024.

[14] Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Michael S Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. *CoRR*, abs/2411.02397, 2024.

[15] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

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

[17] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Selective attention improves transformer. *CoRR*, abs/2410.02703, 2024.

[18] Hengjia Li, Haonan Qiu, Shiwei Zhang, Xiang Wang, Yujie Wei, Zekun Li, Yingya Zhang, Boxi Wu, and Deng Cai. Personalvideo: High id-fidelity video customization without dynamic and semantic degradation. *CoRR*, abs/2411.17048, 2024.

[19] Hengjia Li, Lifan Jiang, Xi Xiao, Tianyang Wang, Hongwei Yi, Boxi Wu, and Deng Cai. Magicid: Hybrid preference optimization for id-consistent and dynamic-preserved video customization. *CoRR*, abs/2503.12689, 2025.

[20] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, Tanghui Jia, Junwu Zhang, Zhenyu Tang, Yatian Pang, Bin She, Cen Yan, Zhiheng Hu, Xiaoyi Dong, Lin Chen, Zhang Pan, Xing Zhou, Shaoling Dong, Yonghong Tian, and Li Yuan. Open-sora plan: Open-source large video generation model. *CoRR*, abs/2412.00131, 2024.

[21] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

[22] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. *CoRR*, abs/2411.19108, 2024.

[23] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[24] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *ICLR*, 2024.

[25] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Mach. Intell. Res.*, 22(4):730–751, 2025.

[26] Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality. *CoRR*, abs/2410.19355, 2024.

[27] Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2363–2372. PMLR, 2017.

[28] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*. IEEE, 2023.

[29] James O'Neill. An overview of neural network compression. *CoRR*, abs/2006.03669, 2020.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[31] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*. IEEE, 2023.

[32] Liang Peng, Boxi Wu, Haoran Cheng, Yibo Zhao, and Xiaofei He. Sudo: Enhancing text-to-image diffusion models with self-supervised direct preference optimization. *CoRR*, abs/2504.14534, 2025.

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

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*. IEEE, 2022.

[35] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

[36] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *ECCV*. Springer, 2024.

[37] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*. OpenReview.net, 2017.

[38] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[39] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*. PMLR, 2023.

[40] Wenhao Sun, Rong-Cheng Tu, Jingyi Liao, Zhao Jin, and Dacheng Tao. Asymrnr: Video diffusion transformers acceleration with asymmetric reduction and restoration. *CoRR*, abs/2412.11706, 2024.

[41] Andreas Veit and Serge J. Belongie. Convolutional networks with adaptive inference graphs. *IJCV*, 128(3): 730–741, 2020.

[42] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[43] Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, et al. Wan: Open and advanced large-scale video generative models. *CoRR*, abs/2503.20314, 2025.

[44] Fu-Yun Wang, Zhaoyang Huang, Alexander William Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, Xiaogang Wang, and Hongsheng Li. Phased consistency models. In *NeurIPS*, 2024.

[45] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogério Schmidt Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, pages 8817–8826. Computer Vision Foundation / IEEE Computer Society, 2018.

[46] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *CoRR*, abs/2502.01776, 2025.

[47] Yifei Xia, Suhan Ling, Fangcheng Fu, Yujie Wang, Huixia Li, Xuefeng Xiao, and Bin Cui. Training-free and adaptive sparse attention for efficient long video generation. *CoRR*, abs/2502.21079, 2025.

[48] Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML*. PMLR, 2023.

[49] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. *CoRR*, abs/2410.10819, 2024.

[50] Guangxuan Xiao, Tianwei Yin, William T. Freeman, Frédo Durand, and Song Han. Fastcomposer: Tuning-free multi-subject image generation with localized attention. *IJCV*, 2025.

[51] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. *CoRR*, abs/2408.06072, 2024.

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

[53] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[54] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *CoRR*, abs/2502.18137, 2025.

[55] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhenghong Liu, and Hao Zhang. Fast video generation with sliding tile attention. *CoRR*, abs/2502.04507, 2025.

[56] Peiyuan Zhang, Runlong Su, Yongqi Chen, Hangliang Ding, William Lin, Kevin Lin, Wei Zhou, You Zhou, Yuzhou Nie, and Hao Zhang. Fastvideo is a lightweight framework for accelerating large video diffusion models. https://github.com/hao-ai-lab/FastVideo, 2025.

[57] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*. IEEE, 2018.

[58] Wentian Zhang, Haozhe Liu, Jinheng Xie, Francesco Faccio, Mike Zheng Shou, and Jürgen Schmidhuber. Cross-attention makes inference cumbersome in text-to-image diffusion models. *CoRR*, abs/2404.02747, 2024.

[59] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. In *NeurIPS*, 2023.

[60] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *CoRR*, abs/2408.12588, 2024.

[61] Yibo Zhao, Liang Peng, Yang Yang, Zekai Luo, Hengjia Li, Yao Chen, Zheng Yang, Xiaofei He, Wei Zhao, Qinglin Lu, Wei Liu, and Boxi Wu. Local conditional controlling for text-to-image diffusion models. In *AAAI*, pages 10492–10500. AAAI Press, 2025.

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 technical contributions highlighted in the abstract and introduction are as follows: (1) VORTA, an end-to-end acceleration framework for Vision DiTs (VDiTs), and (2) Coreset Attention, a sparse attention mechanism designed for efficient modeling of long-range dependencies. Both contributions are thoroughly detailed and empirically validated in the manuscript.

   Guidelines:

   - The answer NA 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 NA 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: The limitations of the current method are discussed in the main manuscript and will be further elaborated in the appendix and supplementary materials. These materials include suitable application scenarios, compatible methods, and failure cases.

   Guidelines:

   - The answer NA 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: [NA]

   Justification: This paper does not include formal results such as lemmas, theorems, corollaries, or proofs.

   Guidelines:

   - The answer NA 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 experimental setup and implementation details are disclosed in the main manuscript. Additional code and supplementary information will be provided in the appendix to facilitate reproducibility. However, even with identical software environments and fixed random seeds, minor variations in results may occur due to differences in hardware conditions.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: As previously noted, the code and open-sourced dependencies (*e.g.*, model weights) will be included in the appendix to support reproducibility.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings, datasets, pretrained model weights, and related implementation details are provided in the main manuscript and supplementary materials.

   Guidelines:

   - The answer NA 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: The primary evaluation metric in this study, running latency, remains consistent across repeated runs. Minor variations may arise from hardware-level factors, which are difficult to isolate with our available computational resources. The benchmark requires over

1,000 H100 GPU hours per run and is a widely accepted metric for the target task. Due to the exceptionally high computational cost, performing statistical significance tests is not feasible.

Guidelines:

- The answer NA 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 experimental settings, datasets, pretrained model weights, and related implementation details are provided in the main manuscript and supplementary materials.

   Guidelines:

   - The answer NA 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: We affirm compliance with the NeurIPS Code of Ethics after thorough review.

   Guidelines:

   - The answer NA 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: We provide a thorough analysis and claim the potential societal impacts in the appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our acceleration method is not directly related to the risks of misuse. References to safeguards for compatible models will be included in the code if available.

Guidelines:

- The answer NA 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: All assets (*e.g.*, code, data, models) used in this paper comply with their respective licenses and are credited.

Guidelines:

- The answer NA 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 code and model weights are released under the CC-BY 4.0 license and are accompanied by a README file included in the supplementary materials.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA 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 (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

VORTA: Efficient Video Diffusion via Routing Sparse Attention

Appendix

Table of Contents

A Methodology and implementation details

A.1 Methodological details

Sliding tile attention. Section 3.2 introduced the sliding tile attention [55], the sparse attention pattern we used to model local interactions. To keep the main text concise, we explained only the 1D case. Here, we present the 3D version of the sliding tile attention mask used in our implementation, as detailed in Algorithm 1.

Bucketed coreset selection (BCS). Section 3.2 also introduced the bucketed coreset selection (BCS) method, designed to reduce the computational overhead of modeling long-range interactions. In the naive setting, computing pairwise similarities across an input sequence of length L incurs a quadratic complexity of $\mathcal{O}(L^2)$. Bolya and Hoffman [4], Sun et al. [40] select a subset of tokens (*e.g.*, 25%) as anchors and computing similarities between these anchors and the remaining tokens. This reduces the number of comparisons but still results in $\mathcal{O}((3L/4) \cdot (L/4)) = \mathcal{O}(L^2)$ complexity.

In contrast, BCS achieves linear complexity $\mathcal{O}(L)$ by employing a bucketing strategy. Each bucket, of size (t, h, w), computes similarities between a central token and the remaining $(thw - 1)$ tokens, yielding a per-bucket cost of $\mathcal{O}(thw)$. With $L/(thw)$ such buckets, the total cost becomes $\mathcal{O}((L/(thw)) \cdot thw) = \mathcal{O}(L)$. No inter-bucket comparisons are performed, and the emperical results in Section 4.1 show that this approach is effective enough in selecting a representative subset of tokens for long-range interactions. Compared to global pairwise methods [4, 40], which require $\mathcal{O}(L^2)$ operations, BCS offers a substantially more efficient $\mathcal{O}(L)$ alternative.

A.2 Implementation details

Implementation details for our VORTA. Besides the implementation introduced in Section 4, we also provide the implementation details for our VORTA model.

For router optimization, we train on the Mixkit dataset [20] for 100 steps using a learning rate of 10^{-2} and a batch size of 4. Training completes in approximately one day using two H100 GPUs.

Algorithm 1 3D sliding tiled attention mask

Input: video size $\mathbf{v} = (F, H, W)$, tile size $\mathbf{t} = (t_F, t_H, t_W)$, window size $\mathbf{w} = (w_F, w_H, w_W)$.

// Total sequence length for self-attention
$L \leftarrow F \times H \times W$
$\mathbf{M} \leftarrow \mathbf{0}_{L \times L}$
// The attention mask between the i-th query and the j-th key
for $i \leftarrow 1$ to L **do**
  for $j \leftarrow 1$ to L **do**
    // Get tile id from token id
    $q_F, q_H, q_W \leftarrow \text{get\_tile\_id\_3d}(i, \mathbf{v}, \mathbf{t})$
    $k_F, k_H, k_W \leftarrow \text{get\_tile\_id\_3d}(j, \mathbf{v}, \mathbf{t})$
    // All queries within the same window share the same tile id as the window center tile id
    $q_F, q_H, q_W \leftarrow \text{get\_window\_center\_id}(q_F, q_H, q_W, \mathbf{w})$
    *// **true** if key tile is within the local window of query tile*
    $m \leftarrow \text{bool}(\text{abs}(q_F - k_F) \leq w_F/2)$
    $m \leftarrow m$ **and** $\text{bool}(\text{abs}(q_H - k_H) \leq w_H/2)$
    $m \leftarrow m$ **and** $\text{bool}(\text{abs}(q_W - k_W) \leq w_W/2)$
    // Save the mask
    $\mathbf{M}[i, j] \leftarrow m$
  end for
end for

Output: sliding tile attention mask $\mathbf{M}$.

The sliding attention branch employs a window size of $(18, 27, 24)$, while the coreset attention branch uses a bucket size of $(2, 3, 2)$ with a coreset ratio of $r_{\text{core}} = 0.5$.

Regarding video configurations during both training and inference, HunyuanVideo [16] utilizes videos with 117 frames at a resolution of 720×1280 (720p), while Wan 2.1 [43] uses 77-frame videos at the same resolution. For rendering, HunyuanVideo outputs videos at 24 frames per second (FPS), whereas Wan 2.1 generates videos at 15 FPS, as specified in their respective repositories.

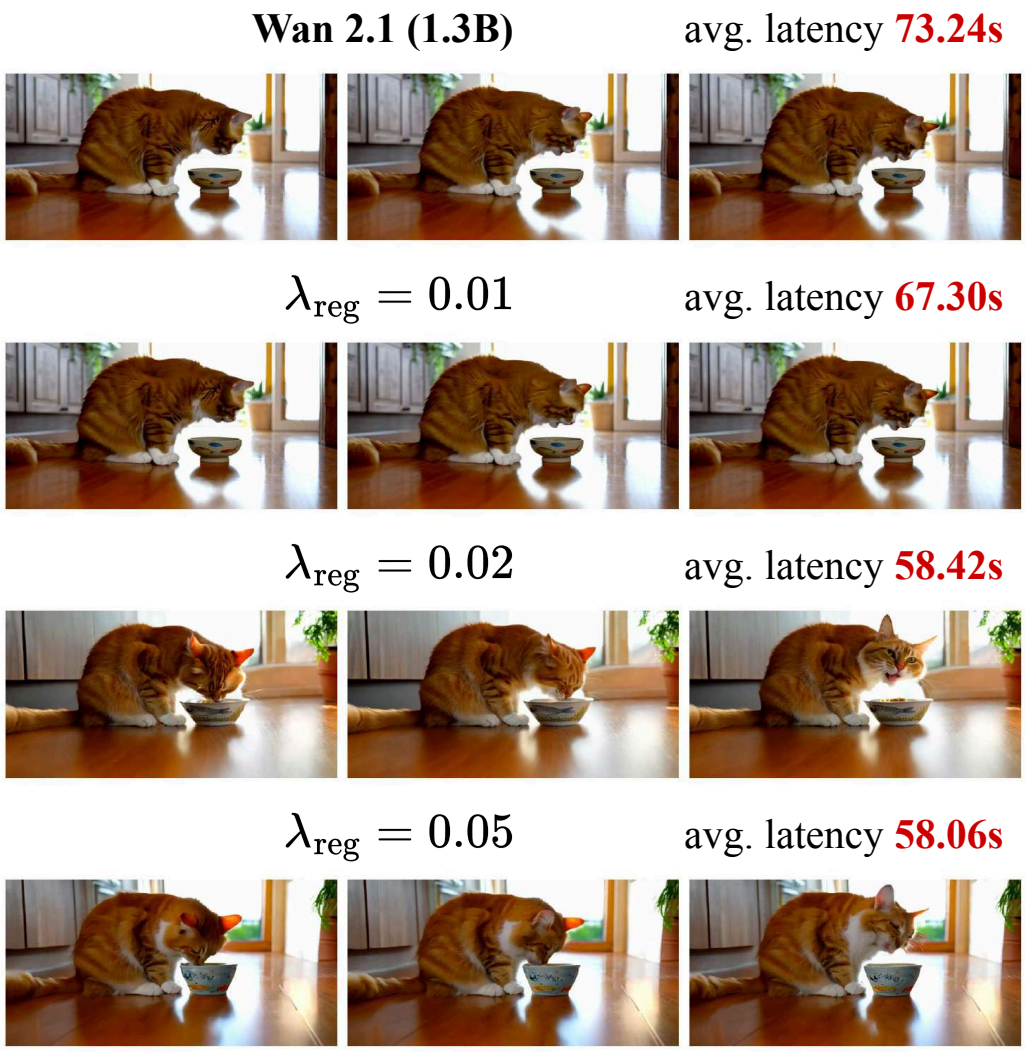
Implementation details for baseline methods. To evaluate efficiency, the PAB [60] is tested on 720p videos from HunyuanVideo [16], aligning with the setup used in other methods. However, processing at this resolution with PAB requires over 80 GB of GPU memory. To address this limitation, we apply sequential CPU offloading [42]. For a fair comparison, we exclude the latency caused by model loading and offloading from the reported results. Despite this, the end-to-end runtime with CPU offloading exceeds 2000 seconds per video, which is substantially slower than the original pretrained model and offers no practical efficiency advantage. The original STA kernel supports video generation only at a fixed resolution of 768×1280, which exceeds the 720p resolution used by the pretrained model. To ensure consistency in evaluation, we reimplemented the kernel using FlexAttention to support 720p video generation.

Evaluation metrics. To evaluate how the generated outputs differ from those of pretrained models, we use LPIPS [57] as a reference metric. However, since the pretrained models do not represent the ground truth, divergence from them does not necessarily indicate degraded performance. Multiple generated outputs may be equally valid, provided they align with the input prompts. LPIPS is only used as a comparative reference. The primary evaluation of video quality and prompt alignment should be based on VBench [11] scores.

The improved performance of acceleration methods. Interestingly, despite using less computation, VORTA slightly outperforms the original pretrained models in terms of VBench score. Similar findings have been reported in other acceleration methods [17, 48]. A possible explanation lies in the redundancies present in overparameterized models, which can introduce marginal negative effects. By pruning these redundancies, these acceleration methods may contribute to slight performance gains, although they are not intended for this purpose.

B Additional experimental results and findings

B.1 Hyperparameters analysis



prompt: *a fluffy orange tabby cat ... eating from a ceramic bowl decorated with fish patterns ...*

Figure 10: Qualitative evaluation of varying the regularization weight λ_{reg}.

Figure 10 shows the effect of varying the regularization weight λ_{reg}. When λ_{reg} is small ($\lambda_{\text{reg}} = 0.01$), the speedup is limited, and in most scenarios, the router tends to select full attention. In contrast, with a large $\lambda_{\text{reg}} = 0.05$, the video exhibits noticeable distortion (*e.g.*, the cat's head in the final frame). A moderate value of $\lambda_{\text{reg}} = 0.02$ achieves a good trade-off between acceleration and output quality.

Figure 11 provides further analysis of alternative pooling strategies in the coreset attention branch. Using average pooling with a $r_{\text{core}} = 50\%$ coreset ratio significantly underperforms compared to BCS pooling, primarily because it lacks a selection mechanism. As the coreset ratio r_{core} increases, the VBench score improves; however, a ratio of $r_{\text{core}} = 50\%$ is sufficient to achieve strong performance. Higher ratios yield marginal performance gains while introducing additional computational overhead during generation.

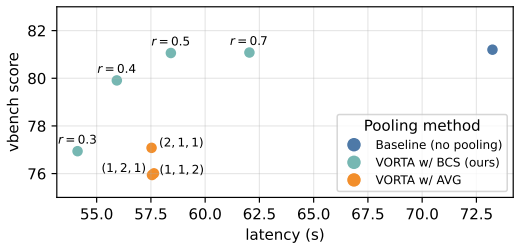


Figure 11: Effect of varying the coreset size in BCS and the kernel size in average pooling.

B.2 Runtime analysis

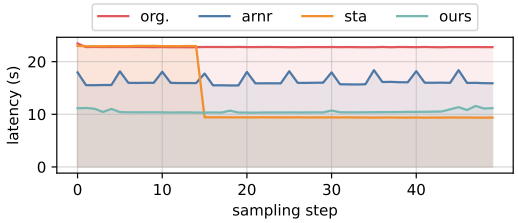


Figure 12: Per-step sampling latency on HunyuanVideo [16]. ARnR [40] yields smaller speedups overall. STA [55] shows no speedup in early timesteps.

Section 4.1 analyzed the average runtime (over diffusion steps) of each component, as shown in Figure 8. Figure 12 further presents the runtime breakdown per diffusion step. Overall, ARnR exhibits relatively high runtime, primarily due to limited acceleration in attention operations. Additionally, its periodic global similarity computation, with $\mathcal{O}(L^2)$ complexity, introduces a bottleneck. This is evident from latency spikes every five steps. In contrast, STA shows significantly higher runtime during the initial 15 steps due to the lack of early-stage acceleration, leading to reduced overall efficiency. Our VORTA achieves near-constant runtime across all steps, demonstrating stable and efficient performance.

B.3 Attention pattern analysis

Figures 13 and 14 show the router gate values for Hunyuan [16] and Wan 2.1 (14B) [43], respectively, as supplementary results for Section 3.1.

B.4 Qualitative comparison

In addition to the qualitative results in Figures 1 and 7 and the quantitative results in Table 1, we present further visualizations for Wan 2.1 (14B) in Figure 15. VORTA inherits the strong performance of the pretrained ViTs while offering a significant speedup.

B.5 VBench dimensional scores

VBench evaluates the generated videos across 16 dimensions. Due to space constraints, we report only three aggregated scores in Table 1, with the complete set of scores provided in Table 4 for completeness.

B.6 Comparison with SVG

We evaluate Spare Video Gen (SVG) [46] on a B200 GPU and compare it with our VORTA, as shown in Table 5. The comparison covers four VBench dimensions for quality and latency, as well as peak memory usage for efficiency. Since both SVG and VORTA focus on attention sparsity, minor non-attention optimizations (*e.g.*operator fusion and quantization) are disabled in SVG to ensure a

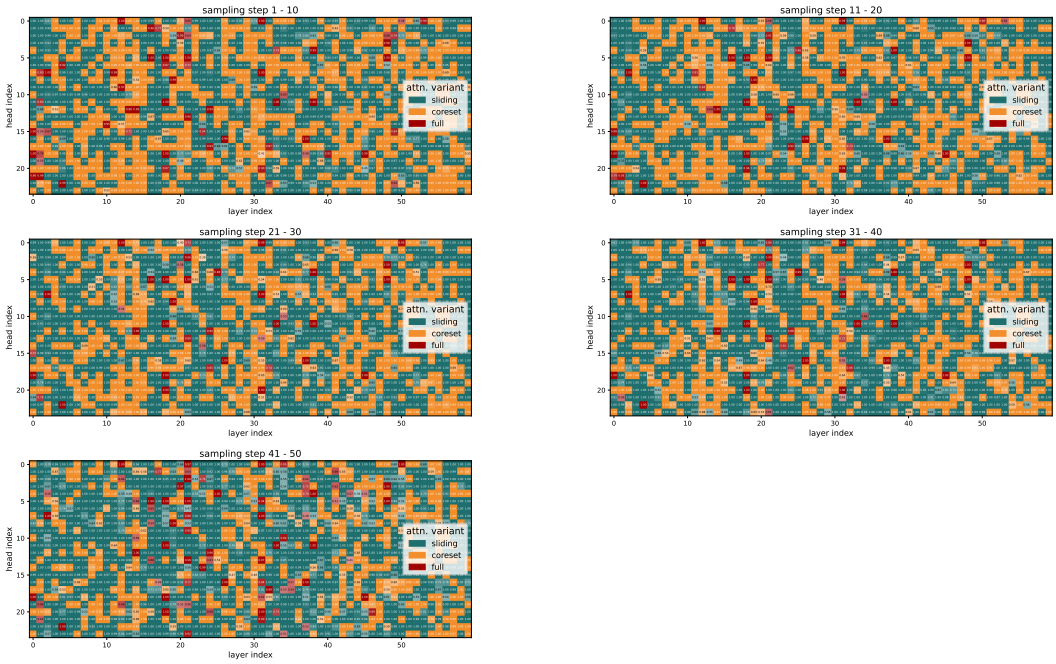


Figure 13: Optimized gate values for HunyuanVideo [16]. The 50 sampling steps are grouped into 5 equal intervals, and the averaged gate value within each interval is reported. Each color corresponds to a distinct attention branch, with color intensity indicating confidence.

fair comparison. We define attention sparsity as the ratio of skipped query-key token multiplications to the total number of query-key multiplications in standard dense attention. A higher sparsity indicates better theoretical computational efficiency.

For both Hunyuan and Wan 2.1 models, the VBench scores are statistically indistinguishable, indicating no significant differences in quality. However, VORTA consistently outperforms SVG in latency and sparsity metrics. Additionally, SVG incurs substantially higher memory usage due to its reliance on online profiling, which limits its applicability in resource-constrained environments.

C Limitations and border impact

C.1 Failure cases

VORTA does not modify the pretrained parameters of VDiTs, and therefore its performance inherently depends on the quality of the underlying pretrained model. As illustrated in Figure 16, when the pretrained VDiTs fail to generate high-quality videos, resulting in distortions or non-physical outputs, VORTA exhibits similar deficiencies. In some cases, it even produces outputs of lower quality than the original VDiTs. This degradation is attributed to computations on erroneous generations, which amplify distortions in the resulting videos.

C.2 Border impact

Generative models pose risks of producing biased, privacy-invasive, or harmful content. While our method accelerates video generation, it may also propagate such risks. It is imperative that researchers, developers, and platform providers actively assess and mitigate these potential harms to promote responsible use.

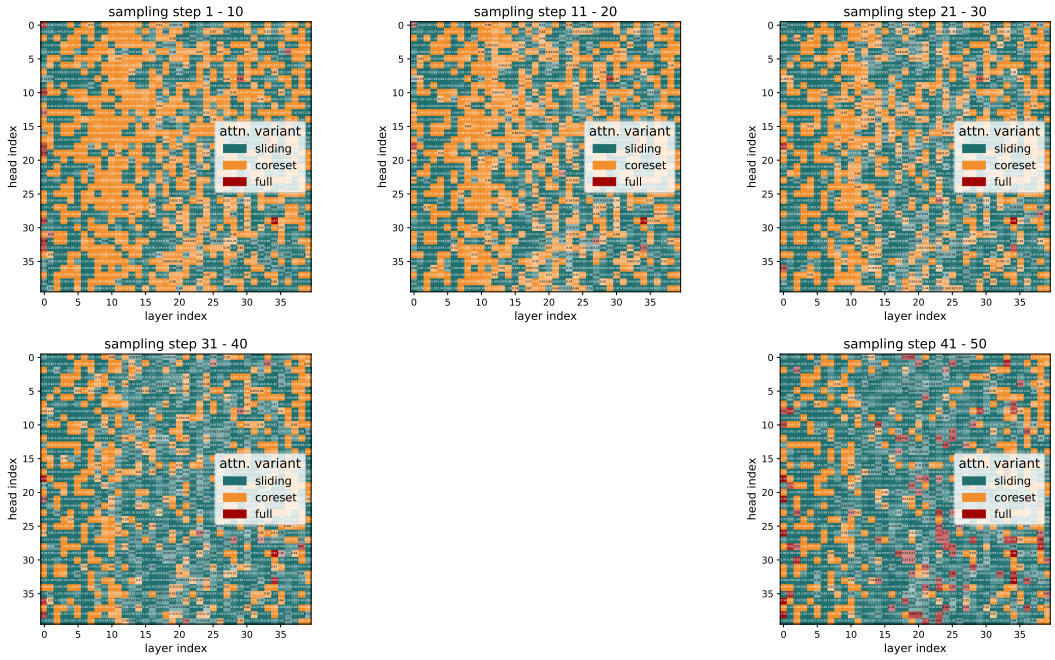


Figure 14: Optimized gate values for Wan 2.1 (14B) [43].

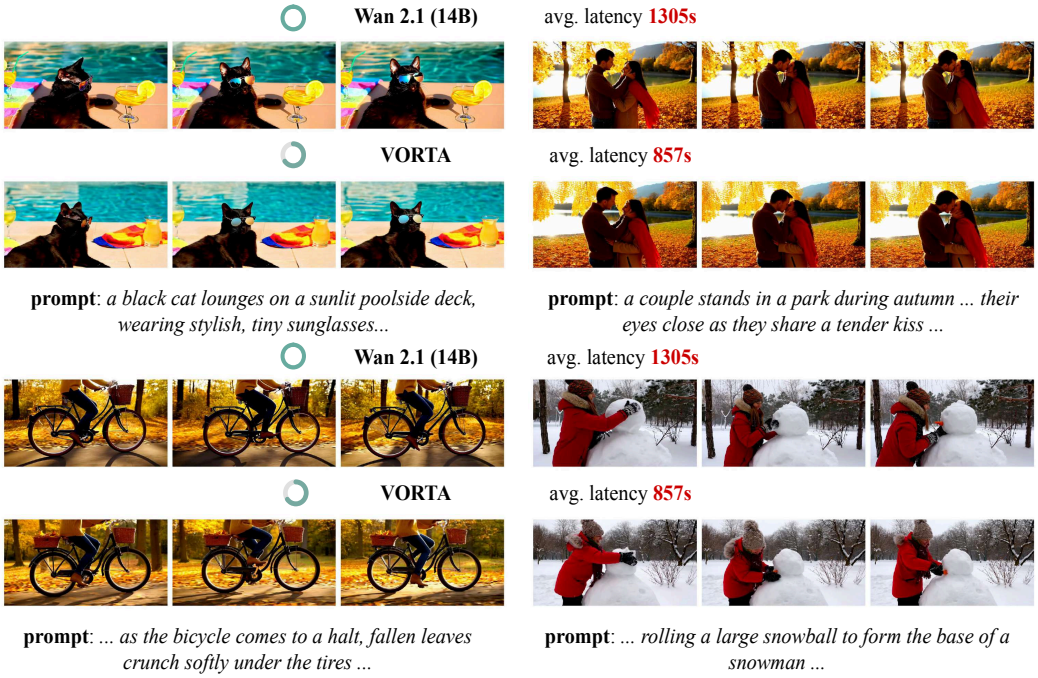


Figure 15: Qualitative comparison on Wan 2.1 (14B) [43].

Table 4: Quantitative results for VBench dimensions [11] for HunyuanVideo [16] and Wan (14B) [43].

Method	Aesthetic Quality ↑	Appearance Style ↑	Background Consistency ↑	Color ↑	Dynamic Degree ↑	Human Action ↑	Imaging Quality ↑	Motion Smoothness ↑
HunyuanVideo	62.05	77.88	93.80	91.35	38.89	96.00	63.33	96.98
+ ARnR	62.58	80.36	93.99	87.54	39.77	93.73	63.21	96.12
+ STA	63.87	79.48	94.47	86.39	39.58	97.00	61.92	96.79
+ PAB	63.53	75.85	94.82	87.92	36.11	98.00	63.40	97.34
+ VORTA	62.33	80.63	94.62	92.13	37.50	97.00	63.16	95.84
+ VORTA & PAB	63.16	80.43	95.22	90.92	36.31	97.50	62.05	96.18
+ PCD	62.69	76.21	94.44	85.94	31.94	94.00	63.08	96.46
+ VORTA & PCD	61.43	76.92	94.87	89.33	35.42	98.00	62.42	95.11
Wan 2.1 (14B)	63.33	82.12	94.55	83.26	37.50	99.00	63.99	91.60
+ VORTA	63.79	83.32	95.28	87.04	39.58	100.00	63.07	92.17

Method	Mutliple Objects ↑	Objects Class ↑	Overall Consistency ↑	Scene ↑	Spatial Relationship ↑	Subject Consistency ↑	Temporal Flickering ↑	Temporal Style ↑
HunyuanVideo	52.21	84.41	74.30	65.59	78.06	90.30	98.57	69.61
+ ARnR	52.09	87.49	69.88	69.21	80.89	90.55	99.28	70.70
+ STA	56.55	87.82	75.01	64.08	80.60	88.12	98.40	69.59
+ PAB	56.86	84.97	74.97	63.91	78.34	90.89	98.61	70.50
+ VORTA	58.00	89.56	74.62	61.87	78.30	92.43	98.47	69.44
+ VORTA & PAB	59.17	89.90	75.82	62.99	77.43	92.27	97.96	70.29
+ PCD	54.04	81.88	75.07	66.03	74.48	90.64	97.37	70.52
+ VORTA & PCD	51.07	85.05	74.93	67.09	73.70	91.34	97.50	70.69
Wan 2.1 (14B)	74.62	86.47	75.49	64.70	79.16	91.21	97.69	71.54
+ VORTA	75.76	88.45	75.09	63.29	80.28	90.76	97.78	70.70

Table 5: Quantitative comparison with SVG [46].

Models	Sparsity (%)	VBench				Latency (s)	Mem. (GB)
		Subject ↑	Consistency ↑	Aesthetic ↑	Imaging ↑		
HunyuanVideo [16]	0.00	90.30	74.30	62.05	63.33	1224.22	47.64
+ SVG [46]	80.00	90.59	**75.11**	**62.73**	63.17	664.48	71.38
+ VORTA	**82.65**	**92.43**	74.62	62.33	63.16	**568.74**	**51.15**
Wan 2.1 (14B) [43]	0.00	91.21	75.49	63.33	63.99	1359.76	41.77
+ SVG [46]	75.00	90.42	74.96	63.53	**64.45**	921.50	65.01
+ VORTA	**81.68**	**90.76**	**75.09**	**63.79**	63.07	**791.02**	**43.97**

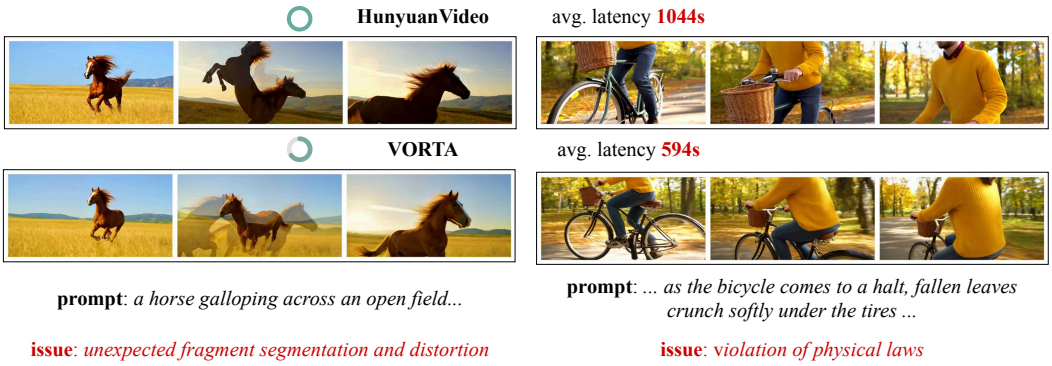


Figure 16: Failure cases. When the pretrained model exhibits distortions or non-physical phenomena, VORTA inherits these issues. We refer the reader to the supplementary video for a more illustrative example.