Faster Video Diffusion with Trainable Sparse Attention

Peiyuan Zhang[1*] **Yongqi Chen**[1*] **Haofeng Huang**[1*‡] **Will Lin**[1]
Zhengzhong Liu[2] **Ion Stoica**[3] **Eric P. Xing**[2] **Hao Zhang**[1]
[1]UC San Diego [2]MBZUAI [3]UC Berkeley

Abstract

Scaling video diffusion transformers (DiTs) is limited by their quadratic 3D attention, even though most of the attention mass concentrates on a small subset of positions. We turn this observation into VSA, a trainable, hardware-efficient sparse attention that replaces full attention at *both* training and inference. In VSA, a lightweight coarse stage pools tokens into tiles and identifies high-weight *critical tokens*; a fine stage computes token-level attention only inside those tiles subjecting to block computing layout to ensure hard efficiency. This leads to a single differentiable kernel that trains end-to-end, requires no post-hoc profiling, and sustains 85% of FlashAttention3 MFU. We perform a large sweep of ablation studies and scaling-law experiments by pretraining DiTs from 60M to 1.4B parameters. VSA reaches a Pareto point that cuts training FLOPS by 2.53× with no drop in diffusion loss. Retrofitting the open-source Wan2.1-1.3B model speeds up attention time by 6× and lowers end-to-end generation time from 31s to 18s with comparable quality, while for the 14B model, end-to-end generation time is reduced from 1274s to 576s. Furthermore, we introduce a preliminary study of Sparse-Distill, the first method to enable sparse attention and distillation concurrently, achieving 50.9x speed up for Wan-1.3B while maintaining quality. These results establish trainable sparse attention as a practical alternative to full attention and a key enabler for further scaling of video diffusion models. Code is available at https://github.com/hao-ai-lab/FastVideo.

1 Introduction

Attention computation is the primary bottleneck when scaling video Diffusion Transformers (DiT) [34, 28]. Even a seemingly short 5-second 720p clip unfolds into more than 100K tokens [29, 20] once flattened as a sequence. Consequently, state-of-the-art video DiTs [20, 35, 43, 26] expend the majority of compute on attention when training on full-resolution, long-sequence data; the trained DiTs remain painfully slow at inference. Fortunately, recent studies [37, 48, 6, 47] reveal an inherent sparsity in DiTs trained using full attention: only a tiny subset of entries in the attention matrix $\mathrm{Softmax}(QK^\top/\sqrt{d})$, which we refer to as *critical tokens*, significantly influence outputs, while the vast majority approach zero. This inherent sparsity calls for developing a native, trainable sparse attention mechanism purpose-built for video DiTs.

Most prior work approaches sparsity as a post-hoc speedup for pretrained DiTs rather than as a first-class training primitive. Sliding Tile Attention (STA) [48] and Sparge Attention [47], for example, begin with a model trained under full attention and then substitute each head with a fixed or profile-derived sparse mask only at inference time [37, 38]. Because the sparsity pattern is decided *after* training, these methods leave the bulk of training cost untouched and introduce a train-test mismatch: the DiT learns parameters in a dense context yet is evaluated in a sparse one. That mismatch caps

[*]Equal contribution. [‡]Work performed during an internship at UC San Diego.

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

 https://doi.org/10.52202/085713-5099

best-case quality at the dense model's ceiling and, in practice, often erodes quality once sparsity is pushed beyond a gentle budget. Consequently, state-of-the-art DiTs still default to quadratic 3D attention despite its prohibitive cost [43, 35, 20, 2, 11].

Designing a trainable sparse attention for video DiTs faces a fundamental chicken-and-egg dilemma: identifying critical token positions traditionally requires computing the full attention matrix, which then erases any computational gains and defeats the purpose of sparse attention. Conversely, resorting to cheap heuristics and not precisely identifying critical tokens may miss high-weight regions and yield suboptimal results. More importantly, any practical attention implementation must honor the block-sparse layouts expected by modern GPU kernels such as Flash Attention (FA) [5] – otherwise theoretical savings do not translate to wall-clock speedup. The central research question is therefore: how can we predict critical tokens accurately, subject to hardware-aligned block structure, without paying the quadratic cost we aim to avoid?

This paper presents VSA (Video Sparse Attention), a trainable, hardware-aligned sparse attention mechanism for video DiTs, drawing inspiration from recent developments in large language models [45, 25, 30]. At the high level, VSA is a hierarchical granular attention, illustrated in Figure 1. The coarse stage first aggregates a cube containing $(4, 4, 4)$ tokens into a single representation to compute cube-to-cube dense attention. Since attention operates on a pooled (short) sequence, it is lightweight, yet *simultaneously* predicts which cube contains critical tokens while modeling global context. A fine stage then applies token-level attention *only* inside the top-$\mathcal{K}$ selected cube. The final output of attention combines both stages through a differentiable gate. Because VSA is end-to-end trainable, it identifies critical tokens not by heuristics, but by *learning from data*. To ensure hardware efficiency, VSA is meticulously designed to map a spatial-temporal cube to a kernel-level tile [1] [48]. This ensures that tokens within a cube are loaded on the same GPU SM and adhere to the block sparse compute layout (§2.2).

One critical parameter in VSA is the tile size. Small tiles let the coarse stage localize critical tokens close to token resolution, while the fine stage attends only to those pinpointed cubes. This sharpens sparsity and improves model quality at the cost of fragmenting work into tiny blocks, which hurts the GPU throughput. In contrast, large tiles boost arithmetic intensity, but the fine stage must then process whole cubes even if only a few tokens inside the cubes matter, thus blurring sparsity (Figure 1 (b)). Another key parameter is whether to inject dedicated local or spatial-temporal patterns into the fine stage. Our systematic ablation studies reveal that an effective VSA configuration combines a global coarse stage with a freely-selectable fine stage. We found that a tile size of 64 and 87.5% attention sparsity achieves performance comparable to full attention while maintaining efficient kernel execution. Furthermore, purposely injecting locality heuristics proved unnecessary. A large sweep of scaling experiments, in which we pretrain video DiTs from scratch (60M to 1.4B parameters with up to 4×10^{21} FLOPS) with 16K sequence length, uncovers a Pareto frontier where VSA achieves near $8\times$ reduction in attention FLOPS and $2.53\times$ reduction in total training FLOPS.

To support VSA's hierarchical sparse attention, we prototype GPU kernel where the coarse stage kernel fuses softmax, Top-$\mathcal{K}$ selection, and block indexing into a single pass, and the fine stage leverages a block-sparse attention kernel based on FA [5]. This leads to our VSA implementation that retains 85% of FA3's MFU [31]. We further retrofit VSA into SoTA open source DiT, Wan2.1 1.3B [35], originally trained with full attention. This integration speeds up attention time by 6x and reduces end-to-end inference latency from 31s to 18s (1.7x) on H100. As a result, VSA enables the first video DiT where attention accounts for only 20% of runtime during both training and inference without quality degradation.

To the best of our knowledge, VSA is the first trainable sparse attention approach that, based on extensive experiments totaling around 90k H200 hours, shows better scaling than full attention on DiTs. Finally, we hope that our explicit ablation of the key parameters (e.g., tile size, critical token prediction, locality prior, sparsity, etc.) will enable more targeted exploration of sparse attention in scaling video DiTs.

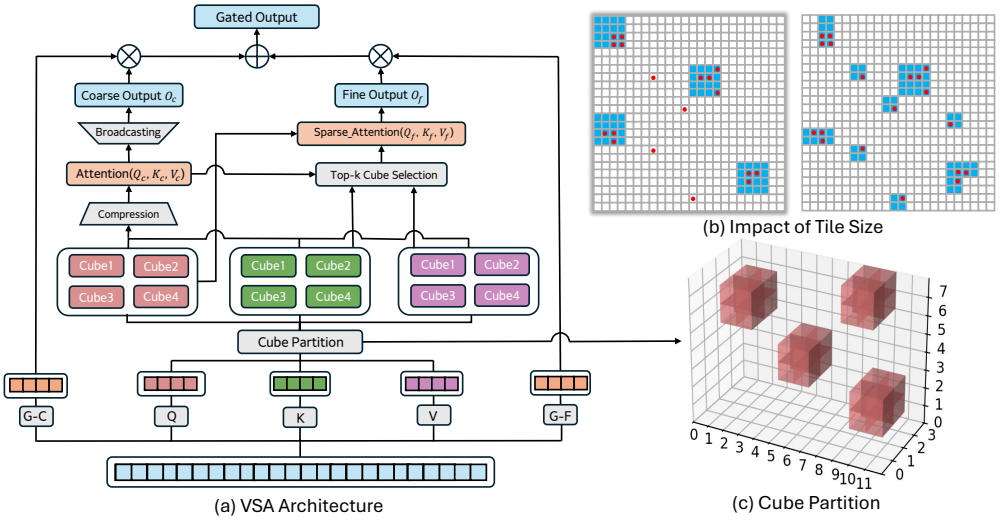


Figure 1: Overview of VSA. (a) VSA introduce a hierarchical attention with sparsity and different granularity (coarse & fine). (b) Larger tile sizes (left) blur attention pattern while small tiles let the coarse stage localize critical tokens close to token resolution. The red dots indicate critical tokens. (c) An illustration of $(2, 2, 2)$ cube partition (in practice we use $(4, 4, 4)$ in VSA).

2 Methods

This section introduces VSA, our sparse attention designed to reduce both training and inference costs of video DiTs. We begin by detailing the design space, key components, and core motivations behind sparse attention in § 2.1. Although the final method is presented in § 2.2, we emphasize that design exploration and ablation studies (deferred to § 3) are central to understanding the effectiveness of VSA. Finally, § 2.3 describes how to adapt VSA to pretrained Video DiTs originally trained with full attention.

2.1 Sparse Attention Design Space

Modern video DiTs use 3D full attention to capture dependencies across the entire video volume. Given a video latent of shape (T, H, W), it is flattened into a 1D sequence of length $L = THW$ by mapping each token location (t, w, h) in the 3D latent to its position n in the 1D sequence following $n = tHW + hW + w$. Then full attention is applied across the entire 1D sequence, allowing each token to interact with all the others. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$ denote the query, key, and value matrices for a single attention head; let $\mathbf{M} \in \{-\infty, 0\}^{L \times L}$ denote the attention mask specifying the allowed connections between each pair of tokens. The attention output $\mathbf{O}$ is then calculated as:

$$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}, \quad \mathbf{A} = \mathrm{Softmax}(\mathbf{S} + \mathbf{M}), \quad \mathbf{O} = \mathbf{A}\mathbf{V}$$

Block Size vs. Hardware Efficiency. In *full attention*, all entries in $\mathbf{M}$ are zero. *Sparse attention* introduces $-\infty$ entries in $\mathbf{M}$ and theoretically reduces the total FLOPS by allowing the computation of the corresponding elements in both $\mathbf{Q}\mathbf{K}^\top$ and $\mathbf{A}\mathbf{V}$ to be skipped. However, modern accelerators are optimized for dense computation, making unstructured sparsity ineffective for real speedup. *Block-sparse attention* [5] addresses this by structuring the sparsity to align with the hardware capabilities. In this approach, the attention mask $\mathbf{M}$ is divided into tiles of size (B_q, B_k) [2], with all entries of

[1]We use the word *block* and *tile* interchangeably in this paper. They refer to a submatrix that a GPU threadblock loads into SRAM when performing matrix multiplication.

[2]Technically (B_q, B_k) can be non-square with different values for B_q and B_k. To keep the notation simple, we assume that $B = B_q = B_k$.

https://doi.org/10.52202/085713-5099

each tile sharing the same value. This enables each tile in a GPU SM to be processed as a dense block or skipped entirely, maximizing hardware efficiency. The tile size B is a key design parameter: smaller tiles allow flexible, fine-grained sparsity but are less efficient on hardware, while larger tiles improve throughput but restrict the model to coarser attention patterns, potentially reducing modeling expressiveness (see Figure 1 (b)). Thus, selecting B involves a tradeoff between expressiveness and efficiency. In practice, we find that modest reductions in speed can be acceptable if they yield significant improvements in generation quality.

Prediction Cost vs. Coverage Quality in Critical Token Selection. Recent studies have shown that the attention score matrix $\mathbf{A}$ is naturally sparse [37, 48, 6, 47], with most values close to zero. This suggests that, by constructing a mask $\mathbf{M}$ that preserves only the high-value regions of $\mathbf{A}$ – the so-called critical tokens – we can closely approximate full attention while significantly reducing computation. A central design choice is how much computation to spend identifying these critical tokens. Computing full attention scores provides the most accurate selection, but largely negates computational savings, as only the $\mathbf{AV}$ operation benefits from sparsity. In contrast, fixed patterns (e.g., windowed or spatiotemporal attention) incur no prediction cost but often miss informative tokens. Inspired by NSA [45] and MoBA [25], we propose a lightweight, trainable, coarse-granular attention module to estimate the locations of critical tokens without fully computing $\mathbf{A}$. The main challenge is to balance prediction accuracy with computational efficiency for practical use in DiT architectures.

Maintaining Global and Local Context in Sparse Attention. A key challenge with sparse attention is its restricted receptive field, which can limit the model's ability to capture global context. One approach to address this is to augment sparse attention with a lightweight global module that captures coarse global signals. Conversely, incorporating local context—motivated by the locality priors commonly used in vision models such as CNNs—can also poteantially enhance feature learning. We empirically ablate both strategies to assess their impact on video generation quality in § 3.1.

2.2 VSA: Video Sparse Attention

VSA employs a cube-based partitioning strategy followed by a two-stage attention mechanism to efficiently process video latents. The method first divides the input video latent into spatially and temporally contiguous cubes, then processes these cubes through a coarse stage that identifies important regions and a fine stage that performs detailed token-level attention within these regions. This design enables efficient computation while maintaining the ability to capture both global and local dependencies in the video data.

Given a video latent with shape (T, H, W), VSA divides it into multiple cubes, each with shape (C_t, C_h, C_w) (Figure 1 (c)). VSA co-designs the sparse attention algorithm and its kernel implementation by mapping each cube in the video latent into a single tile on GPU SM, where the tile size $B = C_t \times C_h \times C_w$. We assume the video latent shape (T, H, W) is an integer multiple of the tile size and define $(N_t, N_h, N_w) = \left(\frac{T}{C_t}, \frac{H}{C_h}, \frac{W}{C_w}\right)$. When flattening the 3D video latent into a 1D sequence, each token at position (t, h, w) is assigned a 1D index n using the following mapping:

$$n = \left(\left\lfloor \frac{t}{C_t} \right\rfloor N_h N_w + \left\lfloor \frac{h}{C_h} \right\rfloor N_w + \left\lfloor \frac{w}{C_w} \right\rfloor\right) B + (t \bmod C_t) C_h C_w + (h \bmod C_h) C_w + (w \bmod C_w).$$

Building on this cube-based partitioning, VSA implements its two-stage attention mechanism to efficiently predict critical token locations *without* computing the full attention matrix $\mathbf{A}$, as shown in Figure 1 (a). In the *coarse stage*, we apply mean pooling over each (C_t, C_h, C_w) cube to obtain cube-level representations, producing $\mathbf{Q_c}, \mathbf{K_c}, \mathbf{V_c} \in \mathbb{R}^{\frac{L}{B} \times d}$. This stage then computes attention scores $\mathbf{A_c} \in \mathbb{R}^{\frac{L}{B} \times \frac{L}{B}}$ and outputs $\mathbf{O_c}$. The attention mask $\mathbf{M}$ is derived by selecting the Top-$\mathcal{K}$ entries per row in $\mathbf{A_c}$ and setting others to $-\infty$, followed by broadcasting to a full-resolution mask of size $L \times L$. This mask naturally conforms to a block-sparse structure because the coarse stage operates on cube-level representations. When broadcasting the mask from $\frac{L}{B} \times \frac{L}{B}$ to $L \times L$, each selected entry in $\mathbf{A_c}$ expands into a $B \times B$ block in $\mathbf{M}$[3]. This block-sparse pattern is crucial for hardware efficiency as it allows the next stage to process attention in contiguous blocks that align with GPU memory access patterns and enable efficient parallel computation.

[3]In practice we do not broadcast to full-resolution mask but only input the selected block index to fine stage.

https://doi.org/10.52202/085713-5099

Next, in the *fine stage*, this mask $\mathbf{M}$ guides fine-grained attention computation for $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$, yielding output $\mathbf{O_f}$. Finally, outputs from both stages are combined to obtain the final output $\mathbf{O}$:

$$\mathbf{O} = \mathbf{O_c} \odot \mathbf{G_c} + \mathbf{O_f} \odot \mathbf{G_f}$$

where $\mathbf{G_c}$ and $\mathbf{G_f}$ are gating vectors obtained from linear projections of the input hidden states. Since the coarse stage introduces negligible computational cost (less than 1% of total FLOPS), the overall sparsity can be approximated by $\frac{\mathcal{K}B}{L}$. However, due to the row-wise Top-$\mathcal{K}$ selection, FlashAttention cannot be directly applied to the coarse stage, resulting in increased latency. § 2.4 discusses how we mitigate this overhead. Appendix B gives a pseudo code implementation of VSA .

In § 3.1 and 3.2, we show that using a smaller B leads to a more expressive sparse attention pattern and improved performance, but comes at the cost of slower attention kernel execution. Setting $B = 64$ with $(C_t, C_h, C_w) = (4, 4, 4)$ provides a favorable trade-off between expressiveness and efficiency. We further demonstrate that combining a coarse stage for global context with a fine stage for token-level sparse attention is both necessary and sufficient—dedicated modules for local context modeling offer minimal additional benefit. Setting $\mathcal{K} = 32$ consistently yields strong performance across a wide range of sequence lengths. VSA adopts these hyperparameters as default.

2.3 Sparse Adaptation & Distillation

VSA is designed to train video DiTs from scratch, reducing both training and inference FLOPS. It can also be adapted to pretrained video DiTs originally trained with full attention. However, directly replacing full attention with VSA leads to unstable training. We hypothesize two main reasons: (1) the gating projection weights $\mathbf{G}$ are not present in the full attention checkpoint and are randomly initialized, and (2) VSA differs significantly from full attention, introducing a coarse stage and sparsity in the fine stage. To address this, we develop an annealing strategy that smoothly transitions the model from full attention to VSA. We initialize the weights of the coarse gate $\mathbf{G_c}$ to zero, and remove the fine gate $\mathbf{G_f}$ (equivalent to $\mathbf{G_f} = \mathbf{1}$). We also initialize the sparsity level by setting $\mathcal{K} = \frac{B}{L}$, effectively making VSA equivalent to full attention at the start of training. As training progresses, we gradually reduce $\mathcal{K}$ to the target sparsity level. Meanwhile, $\mathbf{G_c}$ is updated through training, enabling the model to learn how to balance the contributions of the coarse- and fine stages.

In § 3.3, we demonstrate that Wan-2.1 can be efficiently converted into a sparse-attention model by finetuning with a standard flow-matching loss for only a small number of steps. We further conduct a preliminary study on integrating sparse attention with distillation [44]. In this setup, the student is trained to act simultaneously as a few-step generator and a sparse generator, while the teacher remains unchanged with full attention. Importantly, we preserve the original distillation loss and all hyperparameters, requiring no modifications beyond adapting the student. To our knowledge, VSA is the first sparse attention method shown to be compatible with distillation, whereas prior approaches that exploit diffusion time-step redundancy may fail in the extremely low-step distillation regime.

2.4 Kernel Implementation

VSA requires implementing both forward and backward kernels. We write block-sparse attention kernel with ThunderKittens [32] for fine stage. Despite using a relatively small tile size of 64, our kernel achieves over 85% of FA's MFU (§3.4). The coarse stage, illustrated in Figure 1, requires row-wise Top-$\mathcal{K}$ selection over the cube-level attention matrix. This step necessitates materializing the attention matrix, which precludes direct use of FA-style fused kernels.

One possible workaround is to modify the FA kernel to incorporate in-kernel bitonic sorting for Top-$\mathcal{K}$, thereby avoiding materialization. However, such fusion demands intrusive kernel rewriting and careful tuning. We instead ask: is this complexity necessary? For VSA, the coarse stage operates on $(4, 4, 4)$ cubes, reducing sequence length by $64\times$, e.g., a 100K-token sequence reduces to 1.5K. At this scale, the memory overhead from materialization is negligible. FLOPS-wise, the coarse stage contributes less than 0.2% of total attention compute, and we show in §3.4 that its runtime accounts for only 14%, even when the fine stage is 87.5% sparse. This makes further kernel fusion unnecessary. Nonetheless, we still make some efforts to speed up the coarse stage. Our block-sparse kernel consumes block indices, not binary masks. Therefore, converting the Top-$\mathcal{K}$ mask into index form incurs additional overhead. To mitigate this, we fuse softmax, Top-$\mathcal{K}$ selection, and mask-to-index conversion into a single kernel. This fused kernel reduces coarse stage runtime modestly(§3.4D).

 https://doi.org/10.52202/085713-5099

Exp ID	Case	Loss – Opt	Loss – Over
1	Compress KV	0.15281	0.14282
2	Spatial Temporal	0.13574	0.13034
3	Spatial Full	0.13555	0.12811
4	Strided Window	0.13271	0.12716
5	Full	0.13877	0.12703
6	VSA	**0.13162**	**0.12687**

(a) VSA v.s. other attention.

Exp ID	Case	Loss
7	L	0.13330
8	F (no $\mathbf{O_c}$)	0.13296
9	C & L	0.13220
10	C & F	0.13162
11	C & F & L	0.13194
12	C & F & L & E	**0.13124**
13	C & (F + L)	0.13192

(b) Different attention design.

B	$(C_t,\ C_h,\ C_w)$
256	(4, 8, 8)
128	(4, 8, 4)
64	(4, 4, 4)
16	(2, 4, 2)

(c) B to $(C_t,\ C_h,\ C_w)$.

Exp ID	C Pooling	F Tile	TFLOPS	Loss
14	256x256	256x256	478	0.13375
15	128x128	128x128	444	0.13244
16	64x64	256x256	478	0.13328
17	64x64	64x64	408	0.13162
18	16x64	16x64	181	**0.13155**

(d) Tile size ablation.

Exp ID	Pooling	Loss
19	Conv	0.27787
20	Max	0.13929
21	Avg	**0.13162**

(e) Critical token prediction study.

Table 1: Ablation results for key design parameters of VSA.

3 Experiments

3.1 Ablation Studies

We conduct extensive pretraining experiments to ablate various design choices. Our experiments are based on the Wan2.1 model architecture, a state-of-the-art open-source video DiT. Unless otherwise specified, we train models with 120M parameters from scratch for 4.5×10^{20} FLOPS using video latents of shape $(16, 32, 32)$ from the Vchitect-T2V-Dataverse dataset [10]. We determined 4.5×10^{20} to be compute-optimal for our setup by following established scaling laws [15, 19]: when comparing models of different sizes under fixed compute budgets, we observe that a 120M parameter model with full attention trained at 4.5×10^{20} FLOPS achieves better performance than smaller models (60M) with the same compute budget, while increasing compute beyond this point yields diminishing returns. For VSA and its variants, we set the number of selected KV tiles $\mathcal{K}$ to 32, achieving an attention sparsity of approximately 87.5%. Detailed experimental setup and the rationale behind our experiment design can be found in Appendix C. Key findings from these ablations are presented below.

Data-Dependent Trainable Sparsity Wins Over Fixed-Patterns. We first investigate why full attention predominates despite sparse alternatives. Table 1a shows existing sparse methods (Exp 1-4) outperform full attention (Exp 5) with a compute-optimal training budget (4.5×10^{20} FLOPS), but this advantage reverses with extended training (4×10^{21} FLOPS). VSA (Exp 6) outperforms both previous fixed-pattern methods and full attention. To understand why, in Table 1b(b) we examine two key factors: pattern type and stage contributions. We compare data-dependent patterns against fixed local patterns ("L") that use a $(3, 3, 3)$ window. Simultaneously, we investigate the impact of the coarse stage by including or excluding its output $\mathbf{O_c}$ (denoted as "C") from the final attention output, versus using only the fine stage output (denoted as "F"). Data-dependent patterns consistently outperform fixed patterns, both without the gated residual (Exp 7 vs. 8) and with it (Exp 9 vs. 10), demonstrating the inherent advantage of adaptive sparsity and the gated residual.

Global Information is Necessary; Locality Priors Offer Limited Gains. We tested three approaches for incorporating local context: (1) adding a separate local stage for $(3, 3, 3)$ window attention (Exp 11), (2) explicitly excluding ("E") cubes selected by the local stage from the fine stage (Exp 12), and (3) forcing the fine stage to include local cubes, without a separate local stage (Exp 13). All three variations performed similarly to the simpler C & F architecture, indicating that explicit local modeling provides minimal benefit. VSA therefore adopts the simpler C & F architecture (Exp 10), which effectively captures both global and local information.

Finegrained Attention Map Leads to Better Performance But a Slower Kernel. As analyzed in Section 2.1, the tile size B in VSA is a critical hyperparameter that balances computational efficiency against model performance. It directly affects two key aspects: (1) how accurately critical tokens can be identified through the coarse stage's cube size, and (2) the granularity of attention in the fine stage. Hardware constraints partially dictate this parameter—NVIDIA Hopper GPUs optimize for

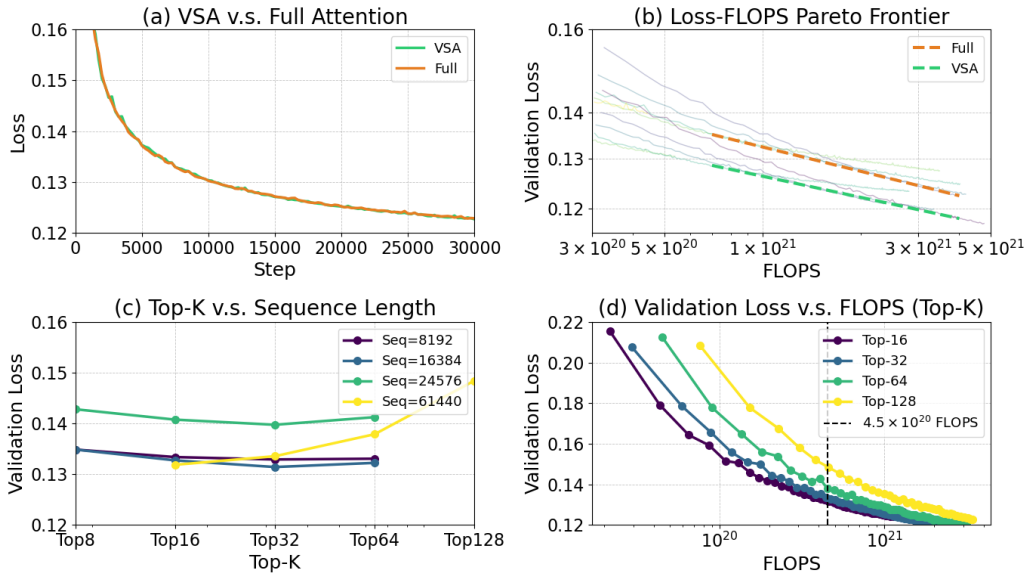


Figure 2: VSA scaling experiments. (a): Video DiT trained with VSA achieves similar loss curve compared to one trained with full attention. (b): VSA consistently produces a better Pareto frontier when scaling model size up to 1.4B. (c) & (d): The optimal Top-$\mathcal{K}$ value (dictating sparsity) depends on both sequence length and training compute. A larger $\mathcal{K}$ is needed for a larger training budget.

matrix multiplications with dimensions divisible by 16, and smaller tiles generally reduce arithmetic intensity. Our benchmarks in Table 1d show that decreasing tile size from 256×256 to 64×16 significantly reduces Model FLOPS Utilization (MFU).

Training with various tile sizes (Table 1c) reveals smaller tiles consistently reduce model loss through finer attention granularity. This improvement stems from the finer granularity allowing the coarse stage to more accurately predict critical-token cubes and the fine stage to focus attention on smaller, more relevant regions. Experiment 16 specifically tests mismatched granularity between stages, confirming that both coarse and fine stage granularity significantly impact performance. Here, the coarse stage used smaller pooling cubes $(C_t, C_h, C_w) = (4, 4, 4)$ (effectively $B = 64$) while the fine stage operated on larger tiles corresponding to $(C_t, C_h, C_w) = (4, 8, 8)$ (effectively $B = 256$). To reconcile the finer granularity of the coarse stage's predicted attention map with the coarser block-sparse attention in the fine stage, we applied an additional $(1, 2, 2)$ average pooling before selecting top-$\mathcal{K}$ entries.

Balancing these findings, we selected 64×64 tiles ($(C_t, C_h, C_w) = (4, 4, 4)$) as our default configuration. While 64×16 tiles offer slightly better performance, they run $2.26\times$ slower (Exp 18 vs. 17), making this tradeoff unfavorable.

Mean Pooling Is Sufficient. We also examined different pooling methods for the coarse stage. Table 1e shows that average pooling outperforms both max pooling and convolutional approaches, with the latter causing training instability.

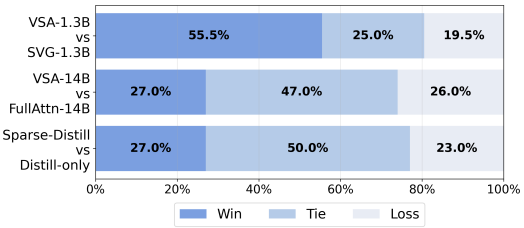
3.2 Scaling Studies

To validate VSA, we pretrained a 410M video DiT with latent shape $(16, 32, 32)$ (16,384 tokens), larger than our 120M ablation models. Figure 2(a) shows VSA achieves nearly identical loss to full attention despite 87.5% sparsity ($\mathcal{K} = 32$ out of 256 cubes), while reducing attention FLOPS by $8\times$ and end-to-end training FLOPS by $2.53\times$. Further scaling experiments from 60M to 1.4B parameters (Figure 2(b)) confirm that VSA consistently produces a better Pareto frontier than full attention. The parallel fitted curves indicate that VSA maintains its $2.53\times$ FLOPS reduction across scales. Each model was trained with compute budgets up to 4×10^{21} FLOPS on 128 H200 GPUs with sequence length of 16K. To our knowledge, VSA is the first trainable sparse attention for video DiTs demonstrating superior performance compared to full attention under rigorous scaling

https://doi.org/10.52202/085713-5099

Model	Qual.	Sem.	Total
Ori-Wan	83.71%	77.98%	82.56%
Full f.t.	84.07%	81.85%	83.63%
VSA f.t.	83.60%	79.47%	82.77%

(a) VSA Wan-1.3B results on VBench. With sparse adaptation, VSA is able to achieve similar score to a full attention counterpart and even slightly outperform the original model.



(b) Top: VSA vs. SVG human evaluation. SVG has a 82.5% attention sparsity with (fp0.03, fl0.025, s0.1). Middle: VSA v.s. full attention with finetuning. Bottom: VSA v.s. full attention with distillation.

evaluation. While we leave comprehensive scaling studies at longer sequence lengths to future work, our fine-tuned Wan 2.1 already shows 1.7× inference speedup at 23K sequence length (§3.3).

An important design question for VSA is determining the optimal sparsity level via the Top-$\mathcal{K}$ parameter. In Figure 2(c), we pretrained 120M models with varying sequence lengths under a fixed 4.5×10^{20} FLOPS budget. Surprisingly, $\mathcal{K} = 32$ performs consistently well across sequence lengths of 8192, 16384, and 24675, but underperforms compared to $\mathcal{K} = 16$ at 61440 sequence length. This contradicts the conventional intuition that longer sequences require higher $\mathcal{K}$ values. Further investigation with increased compute budget at 61440 sequence length (Figure 2(c)) reveals that $\mathcal{K} = 32$ eventually outperforms $\mathcal{K} = 16$ at 1×10^{21} FLOPS, with similar patterns at other lengths. These findings suggest that optimal $\mathcal{K}$ depends on both sequence length and training budget. We hypothesize that the ideal Top-$\mathcal{K}$ increases with available compute, converging to full attention with infinite resources. However, precisely predicting optimal $\mathcal{K}$ given budget, model size, and sequence length remains an open question. One promising direction is to explicitly model sparsity as an additional axis in the scaling law framework [19, 15], alongside model size and total FLOPS. Incorporating inference costs further complicates this analysis, as higher $\mathcal{K}$ values may improve training loss but increase inference overhead. We leave a comprehensive treatment of these tradeoffs to future work.

3.3 Sparse Adaptation & Distillation

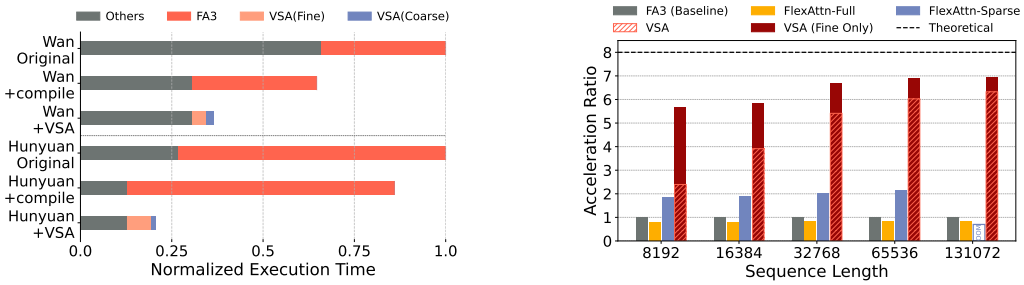
To evaluate VSA's effectiveness in a post-training setup, we finetune Wan2.1-1.3B with VSA on synthetic data generated by Wan-14B with video latent $16 \times 28 \times 52$ (480P). We set $\mathcal{K}$ to 32, corresponding to a 91.2% attention sparsity. As shown in Table 3a, VSA achieves even higher VBench [16] score compared to the original Wan-1.3B. We hypothesize training with synthetic data from a larger model may contribute to this boost. To ensure a fair comparison, we also finetune Wan-1.3B using the same synthetic data. The results show that all models perform closely on VBench, indicating that VSA can retain generation quality despite significant attention sparsity. We additionally compared VSA to SVG [37], a training-free attention sparsification method, under extreme sparsity. Figure 3b Top shows that VSA is preferred even though it has a higher sparsity, demonstrating the effectiveness of training with sparse attention. With VSA, the DiT inference time of Wan-1.3B drops from 31s (full attention with torch compile) to 18s.

To further validate the effectiveness of VSA across different model size and resolution, we scale our study to the Wan-14B model, finetuning on 720P synthetic data (latent $20 \times 48 \times 80$) at 90% sparsity. For this setup, we conduct human preference study on 200 randomly sampled MovieGen prompts [29] to compliment our 1.3B VBench results. As shown in Figure 3b Middle, human evaluation demonstrates that VSA preserves generation quality compared to the official full attention model, indicating that VSA is capable of maintaining high-quality performance at larger model scales. As a pilot study, we further explore whether sparse attention can complement other acceleration techniques, particularly distillation. In this setup, the student model in DMD2 [44] employs VSA while the teacher remains unchanged with full attention. All sparse distillation losses and hyperparameters are kept identical to full-attention distillation, as detailed in C.6. This simple substitution yields a better efficiency (50.9x acceleration) with no quality drop, as shown in Figure 3b Bottom.

https://doi.org/10.52202/085713-5099

3.4 Kernel Performance

As Figure 4b shows, VSA's fine block sparse kernel approaches the theoretical limit with nearly 7× speedup over FlashAttention-3 at long sequence lengths (85% MFU over FA3). Even after accounting for the coarse stage computations, VSA still maintains over 6× speedup. In contrast, FlexAttention [8] with an identical block-sparse mask (64×64 block size) achieves only a 2× speedup. Applying VSA's speedup to Wan-1.3B and Hunyuan brings 2-3×inference speedup, as shown in Figure 4a.



(a) (b)

Figure 4: Kernel benchmarks. (a): Runtime breakdown of a single transformer block for Wan1.3B and Hunyuan. VSA reduces the attention latency by 6×. (b): Speed of VSA with a fixed 87.5% sparsity under various sequence length with head dim 64. VSA approach the theorectical 8× speedup over FA3.

3.5 Inspecting VSA

To gain deeper insight into VSA's mechanism, we inspect the block-sparse attention maps generated by the coarse stage of our finetuned 1.3B model. As illustrated in Figure 5(a-f), the predicted attention patterns are highly dynamic, confirming our hypothesis that effective sparse attention must be data-dependent rather than relying on predefined structures. Even within a single layer, different attention heads often exhibit markedly distinct behaviors. Many observed patterns echo established heuristics, such as local attention focused on tokens near the query (akin to sliding tile attention), or spatial-temporal attention concentrating on tokens within the same frame (d), the same temporal-width plane (e), or the temporal-height plane. Conversely, some patterns deviate from simple heuristics, displaying highly global characteristics (b) or a combination of local and global focus (c).

We quantify the accuracy of critical token prediction calculated as the sum of attention scores within the top-32 cubes selected by the coarse stage. As a baseline, a random selection of 32 cubes from the 386 total (for a (16, 28, 52) latent) captures only 8% of the attention score, as shown by the red plane in Figure 5(e). In stark contrast, VSA maintains a high accuracy rate, consistently achieving at least 60% in most layers and timesteps, and reaching as high as 90% in some instances. This underscores VSA's strong capability in identifying critical tokens. Critically, even if the fine stage misses a small portion of the attention weight, the direct output from the coarse stage can potentially compensate for this. Further examination of Figure 5 (e) reveals systematic variations in prediction accuracy. Accuracy tends to increase monotonically with the timestep. Across transformer layers, however, the accuracy rate exhibits a zig-zag pattern. These accuracy dynamics across layers and timesteps suggest avenues for future optimizations with adaptive Top-$\mathcal{K}$ value.

4 Related Work

Sparse Attention in LLMs. There has been a proliferation of fixed-pattern sparse attention mechanisms in large language models [3, 46, 1, 7, 13]. In practice, however, most LLM training (over 90% of total FLOPs) happens on short sequences ($\leq$32K tokens) under the "train-short, adapt-long" paradigm [40, 23, 12, 42], so sparse attention saw little uptake beyond sliding window attention variants like in Mistral [17]. With LLMs targeting contexts beyond 1M tokens, sparse attention has seen renewed interest. Recent work primarily targets inference-time speedups [39, 49, 18, 41], while the latest methods explore trainable, dynamic sparsity patterns (MoBA [25], NSA [45]) to enable efficient end-to-end training on extreme-length sequences. We draw inspiration from them; However,

https://doi.org/10.52202/085713-5099

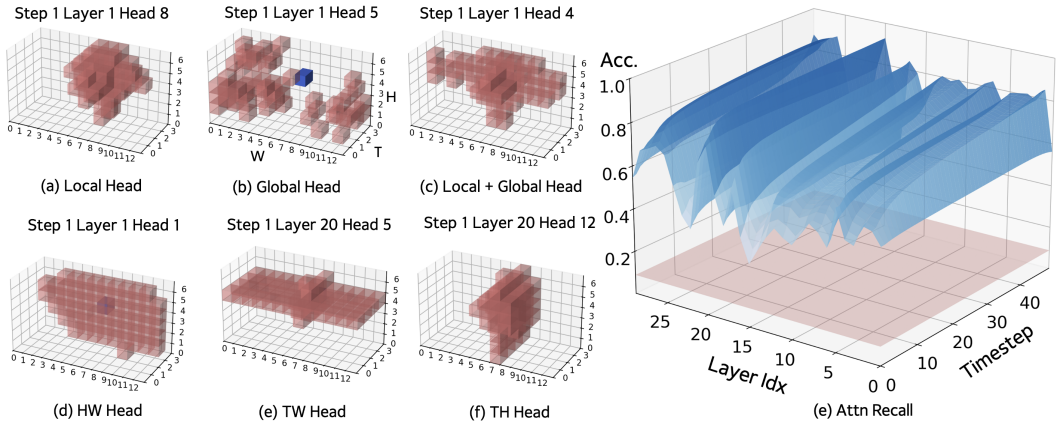


Figure 5: Visualization of the attention pattern of VSA. (a)-(f): VSA dynamically select different cubes to attend, where the blue cube indicates query and red cubes indicated selected key and values.(e): VSA critical-token prediction accuracy.

VSA differs from MoBA by directly contributing the coarse-grained attention output to the final representation and using smaller blocks compatible with efficient block-sparse kernels. Compared to NSA, the nature of video and bidirectional attention avoids grouped query constraints of the attention pattern. We discuss similarities and differences in depth in §E.

Sparse Attention in Video DiTs. Recent work has explored applying sparse attention post-hoc to DiTs pretrained with full attention [48, 6, 47, 37, 14] at inference. However, we argue that the case for trainable sparse attention in video DiTs is both distinct from and *more urgent* than in LLMs. First, video DiT demands far longer sequences, e.g., a 100K-token context yields only 5s video, making scaling inherently more costly than language models. Second, unlike LLMs, where long-context adaptation is a small fraction of total training, state-of-the-art video DiTs [20, 35, 29, 33] dedicate most of their compute budget to full-resolution, long-sequence training. As a result, these models remain bottlenecked by quadratic attention at both training and inference. This calls for trainable sparse attention mechanisms, like VSA, as a core design of video DiTs, not a post-hoc fix. DSV [33] also explores adding sparsity to attention during DiT training, but their multi-stage and profiler-based design may complicate the training pipeline, which we further discuss in §E.

5 Limitation and Conclusion

We present VSA, a trainable and hardware-efficient sparse attention tailored for scaling DiTs. Unlike prior work that applies sparsity post-hoc, VSA jointly learns to predict and apply attention sparsity at training and remains compatible with block-sparse compute layout. VSA currently operates with a fixed cube size of $(4, 4, 4)$, which requires video latent dimensions to be divisible by 4. While this may restrict the set of compatible resolutions, it can be addressed in practice by generating a slightly larger latent and cropping to the target shape. Another open question is how to determine the optimal sparsity level. While our scaling experiments (§3.2) provide preliminary insights, a complete understanding may require extending scaling laws to explicitly account for sparsity, alongside model size and training compute. Across diverse model sizes (60M to 1.4B) and budgets (up to 4×10^{21} FLOPS), we show that VSA matches the performance of full attention at $2.53\times$ lower training cost, and achieves 85% MFU of FA3. When integrated with Wan2.1-1.3B, it reduces end-to-end latency by $1.7\times$. We hope this work establishes trainable sparse attention as a practical and scalable alternative to full attention in further scaling video DiTs.

Acknowledgments

We would like to thank Wei Zhou, Kevin Lin, Matthew Noto, Wenxuan Tan, and Jinzhe Pan for helpful discussion. The work is supported by UCSD HDSI, Nvidia, and a faculty research award from Google. The computing resources were provided by MBZUAI IFM and Nvidia's donation.

References

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

[2] Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, et al. Goku: Flow based video generative foundation models. *arXiv preprint arXiv:2502.04896*, 2025.

[3] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[4] Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. *arXiv preprint arXiv:2304.09151*, 2023.

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

[6] Hangliang Ding, Dacheng Li, Runlong Su, Peiyuan Zhang, Zhijie Deng, Ion Stoica, and Hao Zhang. Efficient-vdit: Efficient video diffusion transformers with attention tile. *arXiv preprint arXiv:2502.06155*, 2025.

[7] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

[8] Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024.

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[10] Weichen Fan, Chenyang Si, Junhao Song, Zhenyu Yang, Yinan He, Long Zhuo, Ziqi Huang, Ziyue Dong, Jingwen He, Dongwei Pan, et al. Vchitect-2.0: Parallel transformer for scaling up video diffusion models. *arXiv preprint arXiv:2501.08453*, 2025.

[11] Genmo. Mochi 1. https://github.com/genmoai/models, 2024.

[12] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[13] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

[14] Ali Hassani, Fengzhe Zhou, Aditya Kane, Jiannan Huang, Chieh-Yun Chen, Min Shi, Steven Walton, Markus Hoehnerbach, Vijay Thakkar, Michael Isaev, et al. Generalized neighborhood attention: Multi-dimensional sparse attention at the speed of light. *arXiv preprint arXiv:2504.16922*, 2025.

[15] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030, 2022.

[16] 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.

 https://doi.org/10.52202/085713-5099

[17] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, et al. Mistral 7b, 2023.

[18] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

[19] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[20] 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.

[21] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, et al. Open-sora plan: Open-source large video generation model. *arXiv preprint arXiv:2412.00131*, 2024.

[22] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.

[23] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[24] 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.

[25] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

[26] Guoqing Ma, Haoyang Huang, Kun Yan, Liangyu Chen, Nan Duan, Shengming Yin, Changyi Wan, Ranchen Ming, Xiaoniu Song, Xing Chen, et al. Step-video-t2v technical report: The practice, challenges, and future of video foundation model. *arXiv preprint arXiv:2502.10248*, 2025.

[27] Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *CoRR*, 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] 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, 2025.

[30] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[31] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

[32] Benjamin F Spector, Simran Arora, Aaryan Singhal, Daniel Y Fu, and Christopher Ré. Thunderkittens: Simple, fast, and adorable ai kernels. *arXiv preprint arXiv:2410.20399*, 2024.

[33] Xin Tan, Yuetao Chen, Yimin Jiang, Xing Chen, Kun Yan, Nan Duan, Yibo Zhu, Daxin Jiang, and Hong Xu. Dsv: Exploiting dynamic sparsity to accelerate large-scale video dit training, 2025.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[36] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *International Journal of Computer Vision*, 133(5):3059–3078, 2025.

[37] 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, 2025.

[38] 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, 2025.

[39] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[40] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4643–4663, 2024.

[41] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

[42] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[43] 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. *CoRR*, 2024.

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

[45] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

[46] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

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

[48] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhenghong Liu, and Hao Zhang. Fast video generation with sliding tile attention, 2025.

[49] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

 https://doi.org/10.52202/085713-5099

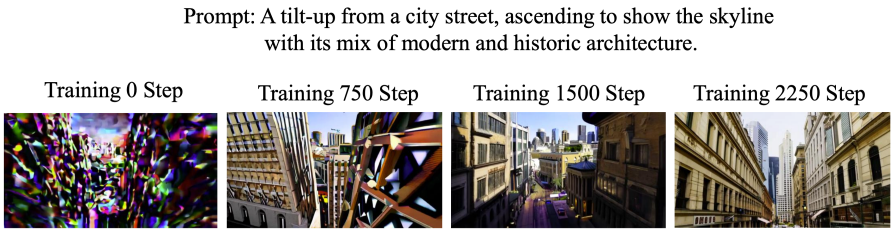
[50] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

[51] Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10323–10333, 2023.

A Qualitative Examples

(a) Qualitative example across training steps

(b) Complete video example at 3000 training steps



(a) Qualitative example across training steps

(b) Complete video example at 3000 training steps

Figure 6: Qualitative examples. In (a), we sample the same middle frame at each step of the video. In (b), we uniformly sample four frames across the video.

We qualitatively illustrate the finetuning process (§2.3§3) of Wan-1.3B in Figure 6. All frames are sampled from validation videos at selected training steps with $\mathcal{K} = 32$. At the start the training, the model exhibits noticeable artifacts when switching from full attention to VSA, reflecting the change in attention structure. As training progresses, the model gradually adapts to the sparse attention mechanism and recovers the ability to generate coherent videos.

B Pseudocode of VSA

We provide a pseudocode in a pytorch-like API for easier understanding of VSA.

```
def tile(x):
    return rearrange(x, "b h (n_t ts_t n_h ts_h n_w ts_w) d -> b h (
        n_t n_h n_w ts_t ts_h ts_w) d",
                     n_t=4, n_h=8, n_w=8, ts_t=4, ts_h=4, ts_w=4)

def untile(x):
    return rearrange(x, "b h (n_t n_h n_w ts_t ts_h ts_w) d -> b h (
        n_t ts_t n_h ts_h n_w ts_w) d",
                     n_t=4, n_h=8, n_w=8, ts_t=4, ts_h=4, ts_w=4)

q, k, v, gate = tile(q), tile(k), tile(v), tile(g)
coarse_attn_gate, fine_attn_gate = gate.chunk(2, dim=1)
```


https://doi.org/10.52202/085713-5099

```
# Coarse stage
B, H, L, D = q.shape
block = 64
topk = 32

q_c = q.view(B, H, L//block, block, D).mean(dim=3)
k_c = k.view(B, H, L//block, block, D).mean(dim=3)
v_c = v.view(B, H, L//block, block, D).mean(dim=3)

score = torch.matmul(q_c, k_c.transpose(-2, -1)) / (D ** 0.5)
score = torch.nn.functional.softmax(score, dim=-1)

output_coarse = torch.matmul(score, v_c)
output_coarse = output_coarse.view(B, H, L//block, 1, D).repeat(1, 1,
    1, block, 1).view(B, H, L, D)

# Keep only top-k blocks
topk_vals, topk_idx = score.topk(topk, dim=-1)
score = torch.zeros_like(score)
score.scatter_(-1, topk_idx, topk_vals)

score = score.view(B, H, L//block, L//block, 1, 1).repeat(1, 1, 1, 1,
    block, block)
score = score.permute(0,1,2,4,5,3).reshape(B, H, L, L)

# Fine stage
QK = torch.matmul(q, k.transpose(-2, -1)) / (D ** 0.5)
QK = QK.masked_fill(attn_mask == 0, float("-inf"))
QK = torch.nn.functional.softmax(QK, dim=-1)
output_fine = torch.matmul(QK, v)

# Combine stages with residual connection
hidden_states = output_coarse * coarse_attn_gate + output_fine *
    fine_attn_gate
hidden_states = untile(hidden_states).transpose(1, 2).flatten(2, 3)
```

Listing 1: Pseudocode of ViLAS in a pytorch-like API with no kernel optimization, assuming cube size (4,4,4) and video size (16,32,32). Note that the tile and untile operation can be moved to the beginning and end of the transformer to avoid calling them for each attention.

C Experimental Details

We document the detailed experiments setups for the results presented in Section 3.

C.1 Model Architecture

We follow the architecture of Wan2.1 [35] for all experiments. Ablation studies are conducted on a 120M-parameter model initialized using the GPT-NeoX scheme, which leads to faster convergence than the default PyTorch initialization. The model adopts the pretrained UMT5-XXL [4] as the text encoder and the pretrained VAE from Wan2.1 for video tokenization. The architecture includes two types of attention: (1) self-attention among video tokens and (2) cross-attention for injecting textual information. Sparse attention is applied only to the self-attention layers. Detailed model configurations are provided in Table 2a.

C.2 Ablation Experiments Setup

We train on long sequences of shape $61 \times 512 \times 512$, motivated by two factors. First, attention dominates the computational cost at this scale, making it the primary bottleneck. Second, sparse attention must be evaluated under long-context settings to demonstrate its effectiveness; short sequences do not present sufficient challenge.

Table 2: Model configuration and training hyperparameters used for the ablation studies.

Model Config	Value
Head Dim	64
FFN Dim	3072
Cross Attn Norm	True
Freq Dim	256
In Channels	16
Out Channels	16
Num Heads	12
Num Layers	12
Patch Size	$1 \times 2 \times 2$
QK Norm	RMS Norm (across heads)
Epsilon	1×10^{-6}
Text Dim	4096

(a) 120M model used for ablation studies.

Hyperparameter	Value
Learning Rate	6×10^{-4}
LR Scheduler	Constant
Warmup Steps	100
Batch Size	1024
Video Latent Shape	$16 \times 32 \times 32$
Sequence Length	16,384
Attention FLOPs Ratio	–
Weight Decay	1×10^{-2}
AdamW Betas	$(0.9,\ 0.95)$
Objective	Flow Matching [24, 22]
Timestep Sampler	LogitNormal$(0.0, 1.0)$ [9]
Total Traning FLOPS	4.5×10^{20}

(b) Training hyperparameters

To establish a strong baseline, we perform a grid search over batch sizes $\{512,\ 1024,\ 2048\}$ and learning rates $\{5 \times 10^{-5},\ 1 \times 10^{-4},\ 2 \times 10^{-4},\ 6 \times 10^{-4}\}$. The best hyperparameters is used for all ablation variants. Training is conducted under a fixed compute budget of 4.5×10^{20} FLOPs, which we find to be sufficient for training a 120M-parameter model – a 120M model with full attention outperforms a 60M model trained with 4×10^{20} FLOPs, indicating FLOPS budget around 4.5×10^{20} is compute-optimal for a 120M model. Each ablation job takes around 10 hours on 64 Nvidia H200 GPU. Full training hyperparameters are provided in Table 2b.

C.3 Baseline Attention Variants

Spatial-Temporal A widely adopted approach in early video generation works, including Open-Sora [50], OpenSora-Plan [21], LaVie [36], and Latte [27]. We alternate between spatial and temporal attention across layers.

Spatial-Full In spatial-temporal attention, the temporal stage can become overly sparse. For example, with a latent shape of $(16, 32, 32)$, the temporal attention accounts for less than 1% of the FLOPs of full 3D attention. To mitigate this, we design a variant with four spatial layers and one full-attention layer every five layers.

Compress KV This variant pools only the key and value tokens using a $2 \times 2 \times 2$ average pooling, reducing attention FLOPs by $8\times$. The query tokens remain at full resolution. This setup mimics the coarse-grained stage of VSA with a smaller pooling size and no pooling on query tokens.

Strided Window Inspired by Swin Transformer, we propose a strided window attention that increases token interaction on top of spatial-temporal attention. Let W_s and W_t denote the spatial and temporal window sizes. For spatial attention, a query attends to all tokens in the same frame and to those in the same temporal window ($W_t = 2$). For temporal attention, a token attends to the same spatial location and to others in the same spatial window ($W_s = 8$).

Conv Pooling Instead of mean pooling for block-level token aggregation, we use a 3D convolution with kernel size and stride of $(4, 4, 4)$ (same as the block size). The output channel dimension matches the head dimension.

C.4 FLOPS Calculation

Elementwise operations such as LayerNorm and RoPE contribute negligibly to the total computational cost in transformers. Following the approximation in [15], we omit these operations and estimate the model FLOPs as $6ND$, where N is the number of model parameters and D is the number of input tokens.

https://doi.org/10.52202/085713-5099

However, for video DiTs trained on long sequences, attention computation becomes a dominant cost. We therefore incorporate the attention FLOPs following the formulation from FlashAttention [5]:

$$\text{FLOPs} = 6ND + 4 \cdot D \cdot S \cdot A \cdot H \cdot 3.5 \cdot L$$

where D is the number of tokens, S is the sequence length, A is the number of attention heads, H is the head dimension, and L is the number of transformer layers. For sparse attention, we adjust the attention portion according to their sparse pattern.

C.5 Sparse Adaptation Setup

To bridge the gap between full and sparse attention, we adopt a sparsity decay schedule that gradually reduces the number of cubes used in attention computation. The model is first trained with full attention for the initial 50 steps, to accommodate the changed resolution and aspect ratio. Thereafter, we decrease the number of attended cubes by 10 (i.e., reduce Top-K by 4) every 50 steps, until reaching the target sparsity level (In our setting Top-$K = 32$). Unlike directly training the model with extremely sparse attention, our progressive decay schedule enables a smooth transition and mitigates training instability.

In the finetuning experiments for Wan-1.3B, we trained on 80,000 synthetically generated videos from Wan-14B, each with a resolution of 448×832 and 61 frames. To accelerate training and reduce memory usage, we preprocessed both the VAE latents and text encodings. Training was conducted on 32 H200 GPUs using DDP as the parallelism strategy. We set the per-GPU batch size to 1, applied a gradient accumulation of 2, and used a learning rate of $1e{-5}$. The training ran for 4,000 steps.

In the finetuning experiments for Wan-14B, we set the final sparsity to 0.9 and trained on 200,000 synthetic videos from Wan-14B, each with a resolution of 768×1280 and 77 frames. Training was conducted on 64 H200 GPUs using DDP as the parallelism strategy. We set the global batch size to 64, and learning rate to $1e{-5}$. The training ran for 4,000 steps.

C.6 Sparse Distillation Setup

Our Sparse Distillation on Wan-1.3B uses 64 H200 with a per-GPU batch size of 1, and runs for 12 hours. We initialize the generator model with the pretrained weights of the base model and then replace the attention module with VSA (sparsity=0.8), while keeping real score and fake score models the same as the base model. All DMD-related hyperparameters are held fixed, including the number of denoising steps, the generator update ratio, and the guidance scale for real-score model. After 4,000 steps training, the 3-step generator achieves a 50.9× reduction in denoising time relative to the baseline model and can generate a 5-second video in approximately 5 seconds on a single H200.

D Coarse Stage Runtime

For shorter sequence lengths, the coarse stage overhead is more pronounced. Our profiling experiments using nsys (Table 3) reveal that Top-$\mathcal{K}$ selection dominates this runtime. Although we fused the kernels for attention scaling, softmax, and Top-$\mathcal{K}$ operations to reduce memory traffic and improve data locality, it only provided modest improvements. Since the coarse stage overhead becomes negligible at longer sequence lengths – our primary target – we did not pursue further optimizations in this work. However, coarse stage acceleration remains an important direction for future research.

Table 3: Coarse stage runtime (µs).

Breakdown	w/o fusion	w/ fusion
QK^T	0.046	0.046
scale	0.060	
softmax	0.095	0.912
topk	0.869	
PV	0.045	0.045

E Discussion and Extended Related Work

VSA builds upon insights from prior work on trainable sparse attention mechanisms in both language modeling [45, 25] and computer vision [51, 33]. This section situates VSA within this landscape, highlighting key similarities and differentiating design choices.

MoBA [25]: VSA shares conceptual similarities with MoBA, particularly in: (1) employing a coarse-grained stage that utilizes mean pooling, akin to MoBA's gating mechanism, and (2) using attention scores from this pooled representation to guide block selection for sparse attention. However, a key divergence lies in the utilization of the coarse-grained stage output. While MoBA employs pooled attention solely for block selection, VSA incorporates the output of its coarse-grained stage directly into the final attention output, potentially enriching global context representation. More critically, MoBA's implementation, which relies on token gathering and variable-length FlashAttention, constrains it to larger tile sizes (e.g., 512). This can limit the granularity of its sparsity patterns and reduce speedup efficacy, especially for sequences like those at 128K. In contrast, VSA is implemented with block-sparse attention leveraging smaller, fixed-size blocks (e.g., 64x64 as per our findings), aiming for a better balance between performance (Table 1d) and practical world-clock speedup (Section 3.1).

NSA [45]: The two-stage (coarse/compress and fine/select) architecture of VSA bears resemblance to NSA's design. However, fundamental differences arise from their target domains. NSA is tailored for causal language model decoding, where typically only a single query token is processed at a time. This necessitates specific strategies like group query attention to enable efficient kernel implementation(NSA can only pool Key-Value pairs). Video DiTs, operating bidirectionally on entire sequences, do not face the same single-query constraint, allowing VSA to apply pooling more broadly and employ distinct sparse patterns for different attention heads without resorting to grouped queries. Furthermore, NSA includes an additional sliding window stage for local information, a component VSA found unnecessary for video generation in our setup (Table 1b).

BiFormer [51]: Similar to BiFormer, VSA utilizes a coarse-grained, tile-to-tile attention mechanism. However, in BiFormer, this coarse attention serves only to derive the dynamic sparse pattern for the subsequent token-to-token attention and does not directly contribute to the final output. Our ablations (Table 1b) indicate that for VSA , the output of the coarse-grained stage is paramount for achieving optimal performance. Additionally, BiFormer's original implementation lacked direct FlashAttention compatibility, impacting its throughput compared to VSA 's design, which is optimized for hardware-aligned block-sparse operations.

DSV [33]: DSV represents pioneering work in exploring trainable sparse attention specifically for video DiT training. Both VSA and DSV aim to reduce the cost of identifying load-bearing regions. DSV achieves this by introducing dedicated low-rank attention predictors with a reduced head dimension, which are trained in a separate, multi-stage process and are not fully end-to-end integrated with the main DiT training. VSA , on the other hand, reduces this cost by performing attention on spatially pooled representations (e.g., from a 4x4x4 cube of tokens) within its coarse-grained stage. Crucially, VSA is designed to be end-to-end trainable, requiring minimal modifications to existing DiT training frameworks, unlike the more complex training system designed for DSV's predictors.

F Broader Impact

VSA aims to make high-quality video generation more accessible by significantly reducing the training and inference cost of video diffusion models. Our work may help democratize video creation tools for a broader range of users and use cases, including education, animation, and independent media production. However, as with many generative technologies, VSA also presents potential risks. In particular, the ability to generate realistic videos at scale may increase the risk of malicious applications, such as generating deepfakes or misleading media content. We emphasize the importance of developing robust detection tools, usage guidelines, and ethical standards in parallel with technical advances to ensure the responsible deployment of such models.

https://doi.org/10.52202/085713-5099

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   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 paper discuss the limitations of the work performed by the authors.

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: No theoretical result.

   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 paper fully disclose all the information needed to reproduce the main experimental results of the paper. The code is opensourced.

   Guidelines:

 https://doi.org/10.52202/085713-5099

- 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: The paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results.

   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 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.

   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: We follow previous work's standard evaluation process to evaluate our model.

   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 paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments.

https://doi.org/10.52202/085713-5099

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: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   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: The paper discuss both potential positive societal impacts and negative societal impacts of the work performed.

    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: The model we release do not provide additional risks than the open-source base model we use.

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

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: New assets introduced in the paper are well documented and is the documentation provided alongside the assets.

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.

https://doi.org/10.52202/085713-5099

- 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.