# NATIVE TOKEN-LEVEL SPARSE ATTENTION FOR LONG-CONTEXT VIDEO PRETRAINING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Modern video generation models built on bidirectional diffusion transformers (DiT) run quadratic attention over space–time tokens. This $O(N^2)$ cost bottlenecks the scaling of video DiT to longer and higher-resolution generation. Sparse attention mitigates the cost by limiting each query to a subset of key–value pairs. Existing sparse attention mechanisms for video DiT mainly apply block-wise selection, which keeps the implementation hardware-friendly and respects the spatio-temporal locality of nearby video tokens. However, block-wise selection is coarse: whenever the relevant context is finer than a block, the attention budget is spent on irrelevant tokens while relevant ones are missed. The resulting gap to full attention is acceptable for accelerating inference, but not for a *native* sparse attention. It is the attention of the model throughout training, so it pays this gap at every update and has to match full attention. In this paper, we propose VTSA (Video Token-level Sparse Attention), a native token-level sparse attention that is lossless in long-context video pretraining. VTSA selects individual key–value tokens with a small indexer distilled from the model's own attention scores. One selection is shared across the query heads that read one key–value head and across a small tile of neighbouring queries. A custom kernel gathers the selected tokens in the forward pass and scatters their gradients in the backward pass. We compare VTSA with two trained sparse-attention baselines, VSA-style block routing and sliding-window attention, in continued pretraining from dense checkpoints. We use two video DiT backbones with different attention architectures, a 5B multi-head video model (Wan2.2-5B) and a 15B grouped-query audio–video model (daVinci-MagiHuman). VTSA matches full attention while each sparse layer attends to 6–15% of the key–value pairs, at 16k and 32k tokens on the 5B backbone and from 16k to 128k tokens on the 15B backbone. Among the tested budgets, block routing matches full attention only at two to four times as many tokens. VTSA opens the door to long-context video pretraining with native token-level sparse attention.

## 1 INTRODUCTION

Modern video generation models (Google DeepMind, 2025; OpenAI, 2025; Wan et al., 2025; Kong et al., 2024) are mainly built on bidirectional diffusion transformers (DiT) (Peebles and Xie, 2023) with full attention: the clip is encoded into a sequence of space–time tokens, and every token attends to every other token. Full attention is quadratic in the sequence length, and a few seconds of footage already unfolds into tens of thousands of tokens, so attention dominates the cost of training and serving longer and higher-resolution video.

Sparse attention is the natural remedy, because for a given query most of the attention mass falls on a small fraction of the key–value pairs, so most of the quadratic computation is spent on weights that are nearly zero. Trainable methods for video DiT, such as VSA (Zhang et al., 2025b), VMoBA (Wu et al., 2025) and BSA (Zhan et al., 2025), select a subset of key–value blocks for each query, and training-free methods sparsify a frozen model at inference time (Xi et al., 2025; Zhang et al., 2025a). Block-wise selection is hardware-friendly but coarse, because the tokens a query needs are often scattered: a fast-moving object crosses block boundaries, and a face occupies a few tokens per frame. When the relevant context is finer than the block, a block-wise selector spends its budget on irrelevant tokens and misses relevant ones (Fig. 1). To quantify this coarseness, we measure the share of a query tile's attention weight that each kind of selection keeps on the released checkpoints of Wan2.2-5B and



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

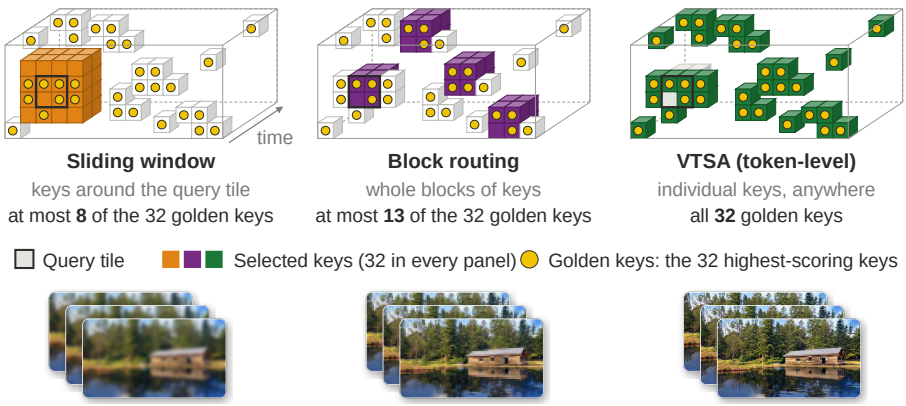



Figure 1: **Three sparse attention designs for video DiT at the same number of selected keys for one query tile.** The gold dots mark the golden keys, the 32 keys with the highest attention scores for the query tile, the same in every panel. Each panel holds 32 selected keys, placed where the design covers the most golden keys: a sliding window selects the keys around the query tile and covers at most 8 of them, block routing selects whole blocks of keys and covers at most 13, and token-level selection can select all 32. The selections are schematic and reduced in size. The frames under each panel sketch the reconstruction the design gives.

the 15B daVinci-MagiHuman at 32k tokens (Table 10, Appendix D). On average, whole 128-token blocks need 1.9 and 2.2 times as many keys as individual tokens to keep 80% of it.

An inference-time accelerator can trade a small loss for speed, because the model it accelerates is still trained with full attention. A *native* sparse attention has no dense model behind it. It is the attention of the model throughout long-context continued pretraining and every later stage, so it pays its gap to full attention at every update and has to match the loss of full attention.

Token-level selection avoids this coarseness but is the hardest granularity to execute efficiently, because an attention kernel reaches full speed only when each block of about 128 keys it loads serves about 128 rows, one row per query and query head. In language models, DeepSeek Sparse Attention (DSA) selects individual tokens and is trained as the attention of DeepSeek-V3.2 throughout long-context continued pretraining (DeepSeek-AI, 2025). DSA is built on multi-head latent attention (MLA), whose query heads all read one latent key–value cache, so the keys selected for one query serve all of its heads, and these heads alone fill the rows. Video DiTs use multi-head attention (MHA) or grouped-query attention (GQA), in which each key–value head is read by only one or a few query heads, so the heads of one query fill too few rows. Whether token-level selection can serve natively as the attention of a video DiT, and what it takes to make it lossless there, therefore remain open questions. To answer both, we propose VTSA (Video Token-level Sparse Attention), a sparse attention mechanism that selects individual key–value tokens. From the dense short-context checkpoint on, VTSA is the video self-attention of the DiT throughout long-context continued pretraining and at inference. VTSA fills the rows by sharing one selection across a small spatio-temporal tile of neighbouring queries and the query heads that read one key–value head. Video is locally redundant, so the queries of such a tile can share one selection. A small indexer distilled from the model's own attention scores selects the tokens for each tile and trains alongside the model, so the selection follows the model as it learns.

We compare VTSA with full attention and block routing on two video DiT backbones with different attention architectures, a 5B multi-head video model (Wan2.2-5B) and a 15B grouped-query audio–video model (daVinci-MagiHuman), and with a sliding window on the 5B backbone. Every variant of a backbone is trained under identical conditions from the same checkpoint, and block routing and VTSA are compared at a matched attended fraction. On the 5B backbone, VTSA is lossless from 15% of the keys at 16k tokens and from 9.4% at 32k, and block routing with the same indexer needs twice that fraction. On the 15B backbone, VTSA is lossless from 15% of the keys in each sparse layer at 16k tokens and from 6% at 32k and 128k, and block routing only from 30%, 25% and 13%.

In summary, our contributions are:

- **VTSA, a native token-level sparse attention for long-context video DiT pretraining.** VTSA selects individual key–value tokens and is the video self-attention of the DiT throughout long-context continued pretraining from the dense checkpoint and at inference.
- **Four components that make token-level selection efficient on MHA and GQA.** Each answers one difficulty of token-level selection: (1) the shared selection fills the rows of the attention kernel, (2) a small tile indexer ranks every key with a scoring pass that costs $1/128$ to $1/24$ of exact scoring, (3) a scattered-KV kernel gathers the selected keys and scatters their gradients back to them, and (4) device-local selection adds no communication under the context parallelism that trains a 15B model at 128k tokens. At the lossless budgets the kernel runs 3.2–8.7$\times$ faster than dense attention at inference, and one forward of the whole model 1.1–2.0$\times$ faster.
- **Lossless at a quarter to a half of the budget of block routing.** At 16k and 32k tokens on the 5B backbone and from 16k to 128k tokens on the 15B backbone, VTSA is lossless from 6–15% of the keys in each sparse layer, and block routing becomes lossless only at two to four times as many keys among the tested budgets. With the exact attention scores as the scorer of both, the remaining gap of block routing is the cost of its granularity. Because VTSA is lossless at the smaller budget, it runs 1.1–1.4$\times$ faster than block routing end to end, each at its lossless budget.

## 2 Video Token-level Sparse Attention (VTSA)

VTSA is a native token-level sparse attention mechanism for video DiT: it is the default video self-attention of the attention blocks during training and inference. It makes token-level selection efficient on the attention architectures of video DiTs, MHA and GQA alike, through four components (Fig. 2): one selection is shared across a small tile of neighbouring queries and across the query heads that read one key–value head, the shared selection keeps the key–value tokens that a small indexer distilled from the model's own attention scores ranks highest, a scattered-KV kernel executes it, and the selection runs on the device that holds the heads it serves.

### 2.1 Notation

Consider one attention layer over a sequence of $N$ tokens, with $H$ query heads of dimension $d$ and $H_{kv}$ key–value heads, so that each key–value head is read by $h_g = H/H_{kv}$ query heads: $h_g$=1 is multi-head attention (MHA), $1 < h_g < H$ is grouped-query attention (GQA), and $h_g$=$H$ is multi-query attention (MQA). Writing $Q_{h,i}$ for query $i$ of head $h$ and $K_{h,j}, V_{h,j}$ for key and value $j$ that head $h$ reads, sparse attention restricts each query to a subset $\mathcal{S}_{i,h} \subseteq \{1, \dots, N\}$ of the keys, scoring and normalising over the subset only,

$$\mathrm{Attn}_{\mathcal{S}}(i,h) \;=\; \sum_{j \in \mathcal{S}_{i,h}} \mathrm{softmax}_{j \in \mathcal{S}_{i,h}}\!\Big(\tfrac{1}{\sqrt{d}}\, Q_{h,i} K_{h,j}^{\top}\Big)\, V_{h,j}, \tag{1}$$

and full attention (Vaswani et al., 2017) is the case $\mathcal{S}_{i,h} = \{1, \dots, N\}$. We fix the number of attended keys per query at $k$ and call $\rho = k/N$ the attended fraction. Fixing $k$, rather than a per-query mass or score threshold, sets the attention FLOPs to a fraction $\rho$ of full attention at any context length. A sparse attention mechanism is then specified by three choices: the *granularity* at which keys are selected, individually ($b$=1, token-level) or in spatio-temporal blocks of up to $b$ tokens ($b$=128 in the block-wise methods we compare against); the *scope* over which one selection is shared, a single (query, head) pair or a tile of $q$ neighbouring queries and a set $\mathcal{G}$ of heads; and the *scoring rule* that ranks the candidate keys.

### 2.2 Challenges toward native token-level sparse attention

Token-level selection is the finest granularity a sparse attention can have, and it is the hardest to execute efficiently, because every efficiency of block-wise selection comes from the block: a block of $b \sim 128$ keys matches the width of a hardware tile, one pooled representative per block leaves $N/b$ candidates to score, and a selected block can be stored contiguously in memory. The bidirectional video DiT has no key–value cache to compress, so its attention is MHA or GQA ($h_g$=1 or 5 on our backbones) rather than the MLA on which DSA (DeepSeek-AI, 2025) is built. This raises four difficulties, and we answer each with one design decision. (1) Only the $h_g$ query heads of one key–value head can read a gathered block of keys, too few rows to fill the hardware tile (§2.3). (2) A

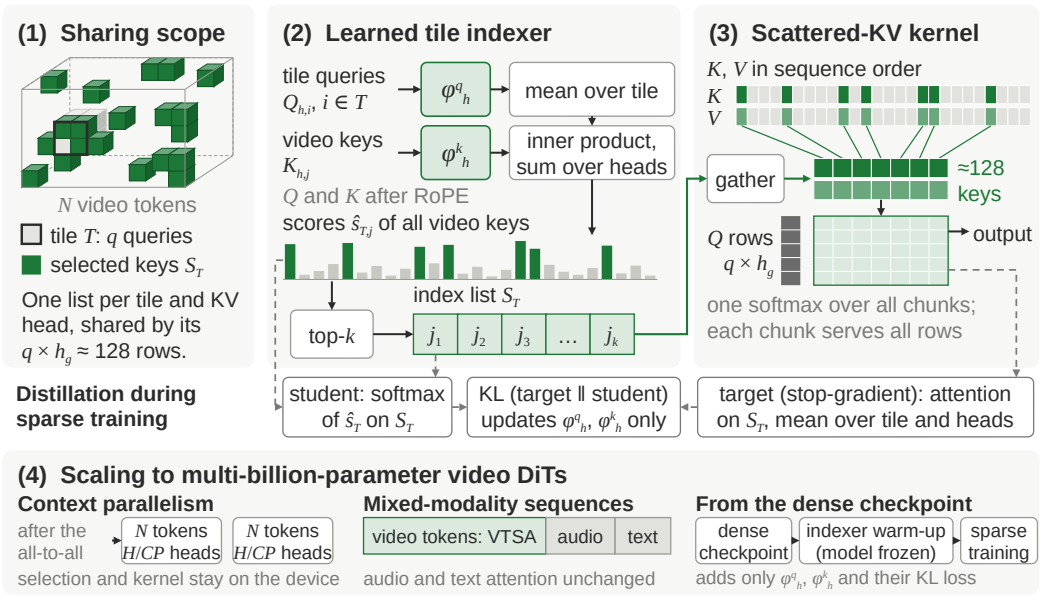



Figure 2: **Overview of VTSA.** (1) One index list $\mathcal{S}_T$ is shared by a tile of $q$ neighbouring queries and the $h_g$ query heads that read one key–value head. (2) The learned tile indexer scores every video key for the tile from the attention's post-RoPE queries and keys and keeps the $k$ highest scores. The dashed paths are its KL distillation during sparse training. (3) The scattered-KV kernel gathers the selected keys and values in chunks of about 128 and runs one online softmax over all chunks for all rows of the tile. (4) The selection runs after the context-parallel all-to-all on the device that holds the heads of the scope, only the video tokens are attended sparsely, and training continues from the released dense checkpoint. The drawing is not to scale.

token-level selector has to rank all $N$ keys (§2.4). (3) The backward pass of FlashAttention walks over keys, and under token-level selection the queries that attend to a key form an irregular list (§2.5). (4) The mechanism has to run inside the system that trains a multi-billion-parameter video DiT, under context parallelism and next to audio and text tokens (§2.6).

### 2.3 Sharing scope: query tiles and device-local heads

One selection serves a scope, a tile $T$ of $q$ neighbouring queries and a set $\mathcal{G}$ of heads, and every query and head in the scope attends to the same $k$ keys. The attention kernel loads a block of $\sim$128 gathered keys and values once and applies it to $\sim$128 rows, one row per (query, head) pair (§2.5). A loaded block belongs to one key–value head and only the $h_g$ query heads that read this head can use it, so the rows are full only when $q$ queries share one selection with $q \times h_g \approx 128$. DSA fills the rows with heads alone, since $h_g{=}H$ under MLA. A video DiT with $h_g{=}5$ or 1 has to fill them with neighbouring queries, and every added query attends to keys selected for its neighbours, so the tile should be as small as the hardware allows; video is locally redundant, so a small spatio-temporal tile can share one selection. We therefore take the smallest tile that fills the hardware, $q \approx 128/h_g$: $2{\times}4{\times}3 = 24$ queries on the $h_g{=}5$ backbone and $8{\times}4{\times}4 = 128$ on the MHA backbone. The heads $\mathcal{G}$ that share a selection are the $h_g$ query heads that read one key–value head, so a selection never crosses a key–value head and stays local under context parallelism (§2.6).

### 2.4 Selection by a learned tile indexer

Token-level selection has to rank every key for every tile. The ideal score of key $j$ is the attention it receives from the tile, averaged over the tile's queries and heads,

$$s_{T,j} \;=\; \frac{1}{q\,|\mathcal{G}|} \sum_{i \in T} \sum_{h \in \mathcal{G}} \Big[ \mathrm{softmax}_{\ell}\Big( \tfrac{1}{\sqrt{d}}\, Q_{h,i} K_{h,\ell}^{\top} \Big) \Big]_{j}, \tag{2}$$

Computing it is a full $QK^\top$ pass, half the forward FLOPs of full attention, so it serves only as the exact scoring rule of the reference variants (§3.2) and as the target of a learned scorer.

For MLA, DeepSeek-V3.2 learns such a scorer, the lightning indexer (DeepSeek-AI, 2025). It scores each query token against one indexer key per token shared by all heads, with a ReLU on each query–key product. Neither choice carries over to a video DiT with MHA or GQA. Its key–value heads attend to different keys. A selection also serves a tile of queries (§2.3), and with a ReLU on each product a tile score costs $q$ query scores.

Our tile indexer therefore keeps one score per key–value head and moves the ReLU from the product to the features. Its inputs are the attention's own queries and keys after rotary embedding. Per query head $h$, two feature maps $\phi_h^q, \phi_h^k : \mathbb{R}^d \to \mathbb{R}^r$ map the queries and the keys to $r = 128$ features, each by two linear layers with a ReLU after each, and key $j$ is scored for the tile by

$$\hat{s}_{T,j} \;=\; \sum_{h \in \mathcal{G}} \Big\langle \frac{1}{|T|} \sum_{i \in T} \phi_h^q\big(Q_{h,i}\big),\; \phi_h^k\big(K_{h,j}\big) \Big\rangle. \tag{3}$$

The score is linear in the query features, so the mean over the tile is taken once in feature space, and a key costs $|\mathcal{G}|\, r$ multiply-adds per tile, $r/(qd)$ of the exact pass: $1/128$ on the MHA backbone and $1/24$ on the GQA backbone. The $k$ keys with the highest $\hat{s}_{T,j}$ form $\mathcal{S}_T$.

We train the indexer toward $s_T$ by the two-stage distillation of the lightning indexer, with the tile as the unit. A 500-step warm-up trains the indexer alone on the frozen dense model, by $\mathrm{KL}(s_T \,\|\, \mathrm{softmax}(\hat{s}_T))$ over all keys. Sparse training then updates the model and the indexer together, and the indexer selects from the first sparse step. Its KL divergence is now taken over the selected keys of every tile, against the attention weights of Eq. 1 averaged over the scope and computed from detached queries and keys, so the sparse stage needs no dense pass. Appendix C gives the training details and ablations.

### 2.5 Scattered-KV attention kernel

A selected set of $k$ tokens is scattered across the sequence, so the FLOPs that selection saves become wall-clock time only when a kernel gathers the keys and values along the index list. We implement this gather in the forward pass and its transpose, a scatter, in the backward pass. In the forward pass one thread block takes the $q \times h_g$ rows of a tile, walks the tile's index list in chunks of $\sim$128 keys, gathers each chunk of keys and values from wherever they sit in the sequence, and runs the online softmax over it, so every gathered chunk serves all rows of the tile.

The backward pass mirrors the forward, because FlashAttention's key-centred backward (Dao, 2024) needs the set of queries that attend to each key, and under token-level selection this set is irregular and unknown until the selection is sorted. A thread block takes the rows of a tile, re-gathers the tile's keys and values in chunks, recomputes the attention probabilities from the saved log-sum-exp, accumulates $dQ$ in registers, and scatters $dK$ and $dV$ to the rows of the selected keys. The rows of a tile reduce their contribution in registers first, so each key receives one contribution per tile that selected it, and the contributions of different tiles are summed in fp32. We implement the forward and the per-tile backward in CuTe (Appendix E).

### 2.6 Scaling to industrial multi-billion-parameter video DiTs

Our 15B backbone trains on sequences of up to 128k tokens with context parallelism, and the base model shards attention by heads: after the Ulysses all-to-all, each device holds the full sequence for $H/\mathrm{CP}$ of the heads. VTSA computes its selection after this all-to-all, on the device that holds the heads of the scope. In our layouts a key–value head and the query heads that read it sit on one device, so every gather and scatter of §2.5 is a local memory access and sparse attention adds no communication to the base model. VTSA replaces the self-attention over the video tokens only, and each backbone continues from its released dense checkpoint, with the indexer's projections and their distillation loss as the only additions.

# 3 EXPERIMENTS

## 3.1 SETTINGS

**Models.** We use two DiT backbones: (1) *Wan2.2-TI2V-5B* (Wan et al., 2025) (Wan2.2-5B below), a 5B DiT model for video generation with MHA (24 query heads, 24 key–value heads, $d = 128$). (2) *daVinci-MagiHuman* (SII-GAIR et al., 2026), a 15B DiT model for audio–video generation with GQA (Ainslie et al., 2023) (40 query heads sharing 8 key–value heads, five per group).

**Training Configuration.** We replace the self-attention over the video tokens with VTSA in every DiT block of Wan2.2 and in every block but the first of daVinci-MagiHuman. daVinci-MagiHuman takes no timestep input (Sun et al., 2025; Tang et al., 2025) and estimates the noise level in its first block by attending over all video tokens, so we keep full attention in that block in every variant, and on this backbone $k$ and $\rho$ refer to the other 39 blocks. The audio and text tokens of daVinci-MagiHuman attend densely, and Wan2.2 reads text through cross-attention. daVinci-MagiHuman is trained on the denoising loss of its video and audio tokens, and we report the video term, since only the video attention is sparsified. We run long-context continued pretraining as in DeepSeek-AI (2025): both models continue from their released dense checkpoints on long-context data with the sparse attention in place.

We train at 16k, 32k and 128k video tokens per clip (12 s at 512p, 6 s at 704p and 10 s at 1088p), each on its own corpus with 1,280 clips held out for validation. Both backbones train at 16k and 32k, and daVinci-MagiHuman also at 128k. Every sparse variant is compared with a full-attention run from the same checkpoint with the same number of updates, data order and seed, so the two differ only in the attention layer. Appendix A gives the corpora and the full configuration.

**Baselines.** We compare full attention, the quality reference, with three sparse variants; block routing and VTSA are compared at matched attended-token budgets, and the sliding window reads a fixed $3 \times 3 \times 3$ block neighbourhood. Every sparse variant attends only to the keys it selects, so the sparse variants differ only in how those keys are chosen: (1) Sliding window: each query block reads the $3 \times 3 \times 3$ blocks around it in the grid of spatio-temporal blocks of 128 tokens that block routing uses, and the box is shifted inward at the edges of the grid so that every query block reads 27 blocks. (2) Block routing in the style of VSA (Zhang et al., 2025b): keys are grouped into spatio-temporal blocks of up to 128 tokens, each block is scored by the attention mass that the scorer of VTSA assigns to its keys, and blocks are gathered in descending score order, the last one truncated so that exactly $k$ keys are attended. As in VTSA, one block list serves the query heads that read one key–value head (§2.3). (3) **VTSA** (§2), token-level selection of the top-$k$ keys. Both VTSA and block routing score the keys with the learned tile indexer, and on Wan2.2-5B at 16k also with the exact attention scores. Under either scorer the two differ only in granularity.

**Evaluation Protocol.** Validation loss is the standard comparison for sparse attention in pretraining and continued pretraining (Yuan et al., 2025; Lu et al., 2025; Zhang et al., 2025b; Zhan et al., 2025), because it is the objective the sparse approximation is trained against. We report the gap $\mathcal{L} - \mathcal{L}^\star$ of the diffusion loss (denoising MSE) on the held-out clips to the full-attention model, averaged over the last ten validation reads of the 5,000-step run, one every 200 steps (the read window, steps 3,200–5,000). We call a sparse variant lossless when this gap is below $4{\times}10^{-4}$, the resolution of the evaluation, which is the size of the difference between two full-attention runs that differ only in their seed (Appendix G).

## 3.2 LOSSLESS ATTENDED FRACTION FROM 16K TO 128K

We sweep $k$ for each design. Fig. 3 plots its gap to full attention against the $\pm 4 \times 10^{-4}$ band, Table 1 lists its lossless budget, the smallest tested $k$ at which the gap is below $4 \times 10^{-4}$, and Appendix B lists the gap at every tested budget.

**The sliding window is far from full attention.** On the MHA backbone at 16k, a window of $3 \times 3 \times 3$ blocks, 3,350 keys per query on average, reads $+58.7 \times 10^{-4}$, against $+8.3 \times 10^{-4}$ for block routing and $+2.5 \times 10^{-4}$ for VTSA at $k = 3{,}072$ (Fig. 3b), so the sweeps below compare token-level selection with block routing only.

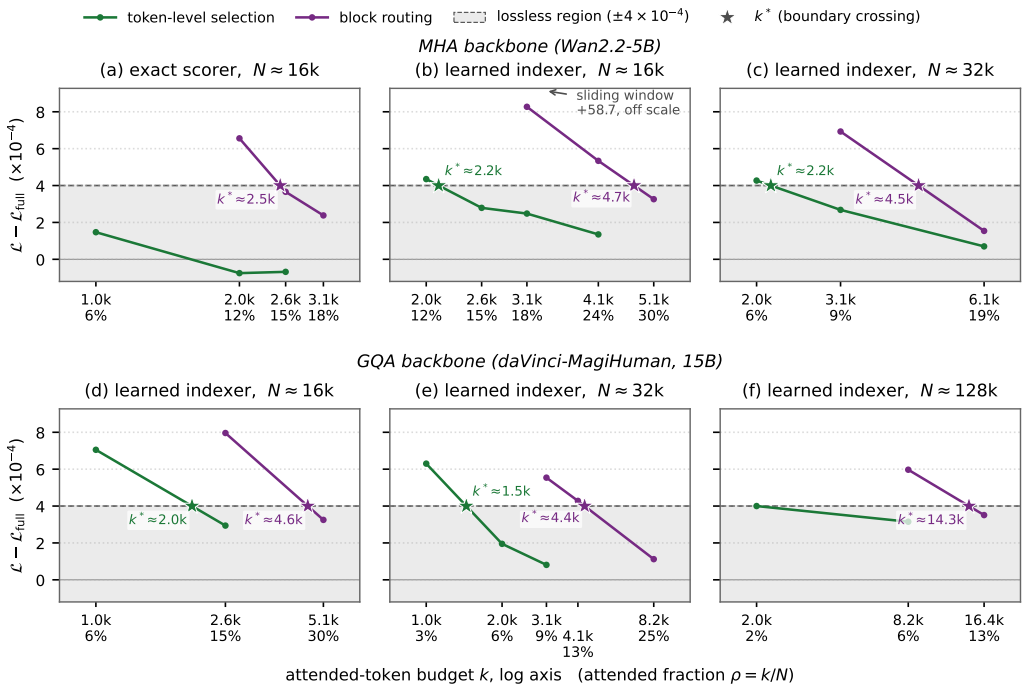



Figure 3: **Gap to full attention against the attended-token budget**. Top row, the MHA backbone (Wan2.2-5B): (a) exact attention scores at $N \approx 16$k, (b, c) the learned tile indexer at $N \approx 16$k and 32k. Bottom row, the GQA backbone (daVinci-MagiHuman, 15B) with the learned tile indexer at (d) $N \approx 16$k, (e) $N \approx 32$k and (f) $N \approx 128$k. The grey band, the lossless region of the legend, is $\pm 4 \times 10^{-4}$ around full attention, and a gap below its upper edge is lossless. A star marks $k^*$, where a line crosses the upper edge between two tested budgets, and a line inside the band at every tested budget has no star. In (f), token-level selection at $k = 2{,}048$ (1.6% of the keys) reads $+4.0 \times 10^{-4}$, on the upper edge, so its line has no crossing and no star. The arrow in (b) is the sliding window.

**With exact scores, token-level selection is lossless from $6\%$ of the keys and block routing from $15\%$, so the gap of block routing is the cost of its granularity.** Token-level selection is lossless at every tested budget down to $k = 1{,}024$ ($+1.5 \times 10^{-4}$), within the scatter of full attention against itself, so attending to few keys costs nothing measurable by itself. Block routing with the same scores reads $+6.6 \times 10^{-4}$ at $k = 2{,}048$ and becomes lossless at $k = 2{,}560$ (Fig. 3a).

**With the learned tile indexer, token-level selection is lossless from $6$–$15\%$ of the keys at every tested context length on both backbones, and block routing, among the tested budgets, from two to four times that fraction.** On the MHA backbone token-level selection becomes lossless at $k = 2{,}560$ at 16k ($+2.8 \times 10^{-4}$) and at $k = 3{,}072$ at 32k ($+2.7 \times 10^{-4}$), while block routing at $k = 3{,}072$ reads $+8.3$ and $+6.9 \times 10^{-4}$ and becomes lossless at $k = 5{,}120$ and $6{,}144$ (Fig. 3b,c). On the GQA backbone it is lossless from $15.0\%$ of the keys at 16k and from $6.3\%$ and $6.4\%$ at 32k and 128k ($+2.9$, $+2.0$ and $+3.2 \times 10^{-4}$). At 128k the smallest tested budget, $k = 2{,}048$ ($1.6\%$ of the keys), reads $+4.0 \times 10^{-4}$, exactly at the resolution, so the lossless budget is the next tested one, $8{,}192$. Block routing is not lossless at $k = 2{,}560$, $4{,}096$ and $8{,}192$ ($+8.0$, $+4.3$ and $+6.0 \times 10^{-4}$) and becomes lossless at two to four times the budget of VTSA (Fig. 3d–f).

**At the lossless budgets VTSA is faster than both full attention and block routing.** Against full attention the kernel runs $3.2$–$8.7\times$ faster, the layer with its indexer $1.2$–$3.3\times$ and the whole forward $1.1$–$2.0\times$ (Table 1). Each larger scale adds work that the attended fraction does not shrink, the indexer scores, whose cost is set by the sequence length and not by $k$, and then the projections and the feed-forward blocks, so the end-to-end speed-up grows with the share of attention in the forward, from $1.10\times$ at 16k to $2.03\times$ at 128k on the 15B backbone. Against block routing VTSA runs $1.6$–$2.8\times$ faster on the kernel and $1.1$–$1.4\times$ end to end. The same kernel serves both designs,

Table 1: **Lossless budget of each design, and the inference speed-up of VTSA at it.** The lossless budget is the smallest tested attended-token budget $k$ (attended fraction $\rho$) at which the gap to full attention is below $4 \times 10^{-4}$, the resolution of the evaluation; the stars of Fig. 3 interpolate between tested budgets, the table lists tested budgets only. Token-level selection with exact scores is below the resolution at every budget of Fig. 3, so its entry is the smallest budget tested. The speed-ups divide the time with full attention (vs full) or with block routing at its lossless budget (vs block) by the time with VTSA at its lossless budget, for one forward of the whole denoiser (end-to-end), one attention layer with its indexer scores, selection and gather (kernel + indexer), and the attention kernel alone (kernel). They are measured at inference in bf16, with VTSA selecting by FlashInfer's top-$k$ kernel (Ye et al., 2025), on one H200 except the 128k end-to-end forward, which runs on eight H200 with context parallelism (Appendix F). "–" is not measured.

| | | | Lossless budget $k$ ($\rho$) | | Speed-up of VTSA at its lossless budget | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | End-to-end | | Kernel + indexer | | Kernel | |
| Backbone | $N$ | Scores | Token-level | Block routing | vs full | vs block | vs full | vs block | vs full | vs block |
| Wan2.2-5B (MHA) | 16k | exact | 1,024 (6%) | 2,560 (15%) | – | – | – | – | – | – |
| Wan2.2-5B (MHA) | 16k | indexer | 2,560 (15%) | 5,120 (30%) | 1.26× | 1.10× | 2.24× | 1.51× | 3.66× | 1.65× |
| Wan2.2-5B (MHA) | 32k | indexer | 3,072 (9.4%) | 6,144 (18.9%) | 1.55× | 1.15× | 3.32× | 1.57× | 5.82× | 1.79× |
| daVinci-MagiHuman (GQA) | 16k | indexer | 2,560 (15.0%) | 5,120 (30.1%) | 1.10× | 1.14× | 1.23× | 1.50× | 3.22× | 1.61× |
| daVinci-MagiHuman (GQA) | 32k | indexer | 2,048 (6.3%) | 8,192 (25.2%) | 1.35× | 1.37× | 2.25× | 2.41× | 6.68× | 2.80× |
| daVinci-MagiHuman (GQA) | 128k | indexer | 8,192 (6.4%) | 16,384 (12.7%) | 2.03× | 1.36× | 2.91× | 1.51× | 8.65× | 1.76× |

Table 2: **Generation quality of the 32k models on the human dimensions of VBench-2.0.** Fidelity is the mean of Anatomy, Clothes and Identity. "Gap" is the validation-loss gap of Table 5 in units of $10^{-4}$. The second full-attention row samples the same weights from three further initial noises.

| Method | $k$ | $\rho$ | Gap | Anatomy↑ | Clothes↑ | Identity↑ | Fidelity↑ |
|---|---|---|---|---|---|---|---|
| *Wan2.2-5B* | | | | | | | |
| Full attention | – | 100% | 0 | 81.73 | 88.57 | 64.38 | 78.23 |
| Full attention, noise seed 2 | – | 100% | – | 79.45 | 92.49 | 60.99 | 77.64 |
| Block routing | 6,144 | 18.9% | +1.5 | 81.37 | 91.37 | 61.10 | 77.95 |
| VTSA | 3,072 | 9.4% | +2.7 | 80.56 | 94.55 | 62.28 | 79.13 |
| *daVinci-MagiHuman* | | | | | | | |
| Full attention | – | 100% | 0 | 82.70 | 89.60 | 85.53 | 85.95 |
| Full attention, noise seed 2 | – | 100% | – | 84.77 | 88.24 | 88.18 | 87.06 |
| Block routing | 8,192 | 25.2% | +1.1 | 82.01 | 89.08 | 84.51 | 85.20 |
| VTSA | 2,048 | 6.3% | +2.0 | 86.09 | 89.00 | 79.68 | 84.92 |

and at the same $k$ the scattered keys of VTSA take at most 5% longer than whole 128-token tiles on the 5B backbone (Appendix F), so the kernel speed-up over block routing comes from the smaller lossless budget of VTSA. On the 15B backbone at 16k and 32k, block routing at its lossless budget is 4% and 1% slower than full attention end to end, because the scoring, selection and packing around its kernel take at least as long as the kernel saves (Appendix F).

### 3.3 Generation Quality

We sample from the 32k models at their lossless budgets. The continued-pretraining corpus is human-centric video, so we score the generated videos on the human dimensions of VBench-2.0 (Zheng et al., 2025): Human Anatomy, Human Clothes, Human Identity and their mean, Human Fidelity (defined in Appendix H). Each model is the average of its five checkpoints from 3,000 to 5,000 updates and generates three clips per prompt (Appendix H). Full attention generates every prompt a second time from three further initial noises, which shows how much each score moves with the initial noise alone.

On Human Fidelity every sparse model lies within 1.1 of one of the two full-attention samples, and the two samples themselves differ by 0.6 and 1.1, so on Human Fidelity the sparse models differ from full attention by about as much as new initial noises alone move it. Fig. 4 compares clips from full attention and from block routing and VTSA at an equal budget on each backbone (more in Figs. 5 and 6). In these clips and those of the galleries block routing drifts over time and loses details. The woman at the snow festival fades into a lens flare, the woman on the ship turns into a cartoon, and a




Figure 4: **Videos generated at 32k by full attention, and by block routing and VTSA at an equal budget.** Block routing and VTSA both select $k = 3{,}072$ keys ($\rho = 9.4\%$), a budget at which VTSA is lossless on both backbones and block routing is not (Table 5). The three models of a backbone get the same prompt. Prompts (abridged): "a woman in jeans and a white T-shirt strolls through a snowy festival" (left) and "a woman reads a farewell letter" (right). Appendix I shows two more prompts on Wan2.2-5B and one more on daVinci-MagiHuman.

second letter appears in the reader's hands. VTSA keeps the person and the objects intact through each clip.

## 4 Related Work

**Sparse attention.** Sparse attention restricts each query to a subset of keys, by fixed patterns (Beltagy et al., 2020; Child et al., 2019; Zaheer et al., 2020), by hashing, clustering or eviction (Kitaev et al., 2020; Roy et al., 2021; Zhang et al., 2023), or by data-dependent selection of key blocks in NSA and MoBA (Yuan et al., 2025; Lu et al., 2025; Tang et al., 2024) and of individual tokens in DeepSeek Sparse Attention (DeepSeek-AI, 2025; Bai et al., 2026; Sun et al., 2026; Zhou et al., 2026; Kwai Keye Team, 2026). A concurrent report compares learned and exact token selectors on frozen LLMs at prefill (Wang et al., 2026). VTSA keeps one selection per key–value head, which reads a lower loss than one selection for all heads (Appendix C), and fills the tile with neighbouring queries instead (§2.3). None of these works trains token-level selection as the attention of a video DiT.

**Sparse attention for video diffusion.** VSA, VMoBA, BSA, MoGA, Veda and Sparse Forcing select blocks or groups of keys (Zhang et al., 2025b; Wu et al., 2025; Zhan et al., 2025; Jia et al., 2025; Han et al., 2026; Xu et al., 2026). Among the tested budgets, block routing needs two to four times the keys of token-level selection to match full attention (§3.2). DSV selects tokens but keeps full attention in layers of low measured sparsity (Tan et al., 2026), SpargeAttention2 fine-tunes under a hybrid top-$k$ and top-$p$ mask (Zhang et al., 2026), and training-free methods sparsify a frozen dense model (Xi et al., 2025; Zhang et al., 2025a; Durvasula et al., 2025; Liu et al., 2026; Chen et al., 2026). These works leave two questions open, which this paper takes up: making token-level selection with a fixed number of keys the attention of a video DiT throughout long-context pretraining, and measuring what block granularity costs at a matched attended fraction.

## 5 Conclusion

We proposed VTSA, a native token-level sparse attention for long-context video pretraining that selects individual key–value tokens through a small tile indexer distilled from the model's own attention. With the exact scores on the 5B backbone at 16k, token-level selection matches full attention from 6% of the keys and block routing only from 15%, so the gap of block routing is set by its granularity. With the learned indexer, token-level selection is lossless at a quarter to a half of the attended fraction that block routing needs among the tested budgets, at every tested context length on both backbones, down to 6% of the keys at 32k and 128k on the 15B backbone.

### AI USE STATEMENT

We used large language models to polish the writing and to help produce some of the analysis figures. The authors reviewed all AI-assisted text and figures and take responsibility for the content of this paper.

### REPRODUCIBILITY STATEMENT

Section 3.1 and Appendix A give, for every comparison, the backbone and the released checkpoint it starts from, the size, resolution and length of the training and validation clips, the optimiser and schedule, the attended-token budgets, the block geometry and the validation protocol. The indexer (Eq. 3) and its training (§2.4, Appendix C), the query tile (§2.3) and the kernel (§2.5, Appendix E) are described in §2. Code for the selection mechanism and the scattered-KV kernel will be released with the camera-ready version.

## REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *arXiv preprint arXiv:2305.13245*.

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. 2026. IndexCache: Accelerating sparse attention via cross-layer index reuse. *arXiv preprint arXiv:2603.12201*.

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

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Naz Sepah, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Dmitriy Serdyuk, Tal Arbel, Christopher Pal, Gaël Varoquaux, and Pascal Vincent. 2021. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems*, volume 3, pages 747–769.

Jiayu Chen, Zhikun Jiang, Maoliang Li, Jiayi Luo, Jiawei Yang, Zihao Zheng, Hengyi Zhang, Guojie Luo, and Xiang Chen. 2026. Token radius attention for efficient video generation. *arXiv preprint arXiv:2608.02504*.

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

Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations*.

DeepSeek-AI. 2025. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models.

Sankeerth Durvasula, Kavya Sreedhar, Zain Moustafa, Suraj Kothawade, Tianlei Pang, Ashish Gondimalla, Suvinay Subramanian, Narges Shahidi, and Nandita Vijaykumar. 2025. FG-Attn: Leveraging fine-grained sparse attention in video diffusion models.

Google DeepMind. 2025. Veo 3 tech report. Technical report.

Shihao Han, Hao Yang, Xinting Hu, Xiaofeng Mei, Yi Jiang, and Xiaojuan Qi. 2026. Veda: Scalable video diffusion via distilled sparse attention. In *International Conference on Machine Learning (ICML)*.

Weinan Jia, Yuning Lu, Mengqi Huang, Hualiang Wang, Binyuan Huang, Nan Chen, Mu Liu, Jidong Jiang, and Zhendong Mao. 2025. MoGA: Mixture-of-groups attention for end-to-end long video generation.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. 2024. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*.

John K. Kruschke. 2018. Rejecting or accepting parameter values in Bayesian estimation. *Advances in Methods and Practices in Psychological Science*, 1(2):270–280.

Kwai Keye Team. 2026. Kwai Keye-VL-2.0 technical report.

Daniël Lakens. 2017. Equivalence tests: A practical primer for $t$ tests, correlations, and meta-analyses. *Social Psychological and Personality Science*, 8(4):355–362.

Daniël Lakens, Anne M. Scheel, and Peder M. Isager. 2018. Equivalence testing for psychological research: A tutorial. *Advances in Methods and Practices in Psychological Science*, 1(2):259–269.

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

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. 2025. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*.

OpenAI. 2025. Sora 2 is here.

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

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

SII-GAIR, Sand.ai, Ethan Chern, Hansi Teng, Hanwen Sun, Hao Wang, Hong Pan, Hongyu Jia, Jiadi Su, Jin Li, Junjie Yu, Lijie Liu, Lingzhi Li, Lyumanshan Ye, Min Hu, Qiangang Wang, Quanwei Qi, Steffi Chern, Tao Bu, Taoran Wang, Teren Xu, Tianning Zhang, Tiantian Mi, Weixian Xu, Wenqiang Zhang, Wentai Zhang, Xianping Yi, Xiaojie Cai, Xiaoyang Kang, Yan Ma, Yixiu Liu, Yunbo Zhang, Yunpeng Huang, Yutong Lin, Zewei Tao, Zhaoliang Liu, Zheng Zhang, Zhiyao Cen, Zhixuan Yu, Zhongshu Wang, Zhulin Hu, Zijin Zhou, Zinan Guo, Yue Cao, and Pengfei Liu. 2026. Speed by simplicity: A single-stream architecture for fast audio-video generative foundation model.

Qiao Sun, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. 2025. Is noise conditioning necessary for denoising generative models? In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. ArXiv:2502.13129.

Yutao Sun, Yanqi Zhang, Li Dong, Jianyong Wang, and Furu Wei. 2026. You only index once: Cross-layer sparse attention with shared routing. *arXiv preprint arXiv:2606.06467*.

Xin Tan, Yuetao Chen, Yimin Jiang, Xing Chen, Kun Yan, Nan Duan, Yibo Zhu, Daxin Jiang, and Hong Xu. 2026. DSV: Exploiting dynamic sparsity to accelerate large-scale video DiT training. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ArXiv:2502.07590.

Bingda Tang, Boyang Zheng, Xichen Pan, Sayak Paul, and Saining Xie. 2025. Exploring the deep fusion of large language models and diffusion transformers for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28586–28595. ArXiv:2505.10046.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. In *International Conference on Machine Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Hongxing Wang, Harenome Razanajato, Zhen Zhang, Yujie Yuan, and Hongsheng Liu. 2026. How much dense attention is necessary? oracle-guided sparse prefill for full/gqa layers in hybrid long-context models. Technical report, first release.

Jianzong Wu, Liang Hou, Haotian Yang, Xin Tao, Ye Tian, Pengfei Wan, Di Zhang, and Yunhai Tong. 2025. VMoBA: Mixture-of-block attention for video diffusion models.

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. 2025. Sparse VideoGen: Accelerating video diffusion transformers with spatial-temporal sparsity.

Boxun Xu, Yuming Du, Zichang Liu, Siyu Yang, Ziyang Jiang, Siqi Yan, Rajasi Saha, Albert Pumarola, Wenchen Wang, and Peng Li. 2026. Sparse forcing: Native trainable sparse attention for real-time autoregressive diffusion video generation. *arXiv preprint arXiv:2604.21221*.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. 2025. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. In *Proceedings of Machine Learning and Systems (MLSys)*. ArXiv:2501.01005.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Chenlu Zhan, Wen Li, Chuyu Shen, Jun Zhang, Suhui Wu, and Hao Zhang. 2025. Bidirectional sparse attention for faster video diffusion training.

Jintao Zhang, Kai Jiang, Chendong Xiang, Weiqi Feng, Yuezhou Hu, Haocheng Xi, Jianfei Chen, and Jun Zhu. 2026. SpargeAttention2: Trainable sparse attention via hybrid top-k+top-p masking and distillation fine-tuning. *arXiv preprint arXiv:2602.13515*.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. 2025a. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *International Conference on Machine Learning (ICML)*.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. 2025b. VSA: Faster video diffusion with trainable sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. 2025. VBench-2.0: Advancing Video Generation Benchmark Suite for Intrinsic Faithfulness. *arXiv preprint arXiv:2503.21755*.

Yanke Zhou, Yiduo Li, Hanlin Tang, Maohua Li, Kan Liu, Tao Lan, Lin Qu, Yuan Yao, and Xiaoxing Ma. 2026. Full attention strikes back: Transferring full attention into sparse within hundred training steps. *arXiv preprint arXiv:2605.16928*.

# A Training Configuration Details

Table 3 lists the configuration that each backbone keeps at every context length, and Table 4 the data and schedule of each setting (§3.1).

Table 3: **Training configuration of each backbone.**

| | daVinci-MagiHuman (15B, GQA) | Wan2.2-5B (5B, MHA) |
|---|---|---|
| Initialisation | released checkpoint | released checkpoint |
| Attended tokens per query $k$ | swept (Table 5) | swept (Table 5) |
| Full attention kept in | the first of 40 DiT blocks | no block |
| Block size of block-wise baselines | $8\times4\times4 = 128$ tokens | same |
| VTSA query tile | $2\times4\times3 = 24$ | $8\times4\times4 = 128$ |
| Heads sharing one selection | 5, one key–value head | 1 (one selection per head) |
| Scoring rule (§2.4) | learned tile indexer, rank 128 (Eq. 3) | learned tile indexer, rank 128 (Eq. 3); exact scoring (Eq. 2) in the reference variants |
| Caption length (text tokens) | 800 | 512 |
| Optimiser | Muon, constant lr $10^{-4}$, weight decay 0.1 | Muon, constant lr $5\times10^{-5}$, weight decay 0.1 |
| Gradient clipping | 1.0 | 1.0 |
| Validation | 1,280 held-out clips, every 200 steps | same |

Table 4: **Data and schedule of each setting.** CP is the context-parallel degree.

| Backbone | Context | Training clips | Clip format | Video tokens | Batch | Steps | Hardware |
|---|---|---|---|---|---|---|---|
| Wan2.2-5B | 16k | 71,680 | 512×896 and 896×512, 149 frames, 12.5 fps | 17,024 | 32 | 5,000 | 8×H200 |
| Wan2.2-5B | 32k | 64,000 | 704×1,280, 145 frames, 25 fps | 32,560 | 16 | 5,000 | 8×H200 |
| daVinci-MagiHuman | 16k | 71,680 | as Wan2.2 at 16k | 17,024 | 32 | 5,000 | 32×H200, CP 8 |
| daVinci-MagiHuman | 32k | 64,000 | as Wan2.2 at 32k | 32,560 | 32 | 5,000 | 32×H200, CP 8 |
| daVinci-MagiHuman | 128k | 140,099 | 1,088×1,920, 249 frames, 25 fps | 128,520 | 8 | 5,000 | 32×H200, CP 8 |

# B Validation Gaps at Every Tested Budget

Table 5 lists the gap to full attention at every budget of Fig. 3 (§3.2); the figure plots them and Table 1 reads the lossless budgets from them.

# C Design of the Learned Tile Indexer

Token-level selection ranks all $N$ keys for every tile. The exact tile score of Eq. 2 needs a full $QK^\top$ pass, half the forward FLOPs of full attention, so a selection made with it costs more than half of full attention at any attended fraction. The scorer of VTSA therefore has to meet two requirements. It has to cost a small fraction of the exact pass. Its top-$k$ list has to keep almost all of the attention that the exact top-$k$ keeps. We derive the learned tile indexer from these requirements in three steps, which fix the kind of scorer, its scoring form and its capacity. Unless stated otherwise, the design experiments run on the MHA backbone at 16k with $k = 2{,}048$.

**A learned scorer.** A hand-built summary of the tile's queries loses far more accuracy than a learned scorer of the same cost (Table 6). A hand-built summary keeps a few representative queries of the tile and scores every key exactly against them. The mean query, which is how VSA summarises the queries of a cube (Zhang et al., 2025b), costs $1/128$ of the exact pass and reads $0.0180$ above exact selection in validation loss. A learned indexer of the same cost reads $0.0008$ above it. Every other summary we test is also further from exact selection than the learned indexer, including the exact scores of 64 of the 128 queries, which cost 64 times as much. We therefore learn the scorer and train it toward the exact score.

**Scoring form.** Our indexer has two properties. First, each key–value head gets its own list of keys, instead of one list for all heads. One list for all heads would have to cover the important keys of every head with the same $k$ keys, so each head would get fewer of its own important keys. Table 7 compares the two choices. With one list for all heads the validation loss is higher, a little for token-level selection and more for block routing. Second, scoring a tile costs as much as scoring one query. We apply the ReLU to the features of each query and key, before their product. The score is then a plain dot product, so the mean score over the queries of a tile equals the score of their mean feature. We

Table 5: **Gap to full attention at each budget of Fig. 3.** Gap in units of $10^{-4}$, the mean over ten validation reads, one every 200 steps, minus full attention at the same steps; bold marks gaps below $4 \times 10^{-4}$, the resolution of the evaluation. MHA backbone (Wan2.2-5B; Fig. 3a–c): every variant trains for 5,000 steps (window 3,200–5,000), and the sliding window, $3 \times 3 \times 3$ blocks (3,168–3,456 keys per query, 3,350 on average), is the arrow in Fig. 3b. GQA backbone (daVinci-MagiHuman, 15B; Fig. 3d–f): the 16k clips carry 17,024 video tokens, the 32k clips 32,560 and the 128k clips 128,520; every variant trains for 5,000 updates on 32 H200 (window 3,200–5,000), keeps full attention in the first of the 40 DiT blocks and selects with the rank-128 tile indexer.

| Backbone | Scores | $N$ | Selection | $k$ | $\rho$ | Gap |
|---|---|---|---|---|---|---|
| MHA | exact | 16k | token | 1,024 | 6% | **+1.5** |
| MHA | exact | 16k | token | 2,048 | 12% | **−0.8** |
| MHA | exact | 16k | token | 2,560 | 15% | **−0.7** |
| MHA | exact | 16k | block | 2,048 | 12% | +6.6 |
| MHA | exact | 16k | block | 2,560 | 15% | **+3.7** |
| MHA | exact | 16k | block | 3,072 | 18% | **+2.4** |
| MHA | indexer | 16k | token | 2,048 | 12% | +4.4 |
| MHA | indexer | 16k | token | 2,560 | 15% | **+2.8** |
| MHA | indexer | 16k | token | 3,072 | 18% | **+2.5** |
| MHA | indexer | 16k | token | 4,096 | 24% | **+1.4** |
| MHA | indexer | 16k | block | 3,072 | 18% | +8.3 |
| MHA | indexer | 16k | block | 4,096 | 24% | +5.3 |
| MHA | indexer | 16k | block | 5,120 | 30% | **+3.3** |
| MHA | – | 16k | window | 3,350 | 20% | +58.7 |
| MHA | indexer | 32k | token | 2,048 | 6.3% | +4.3 |
| MHA | indexer | 32k | token | 3,072 | 9.4% | **+2.7** |
| MHA | indexer | 32k | token | 6,144 | 18.9% | **+0.7** |
| MHA | indexer | 32k | block | 3,072 | 9.4% | +6.9 |
| MHA | indexer | 32k | block | 6,144 | 18.9% | **+1.5** |
| GQA | indexer | 16k | block | 2,560 | 15.0% | +8.0 |
| GQA | indexer | 16k | block | 5,120 | 30.1% | **+3.3** |
| GQA | indexer | 16k | token | 1,024 | 6.0% | +7.1 |
| GQA | indexer | 16k | token | 2,560 | 15.0% | **+2.9** |
| GQA | indexer | 32k | block | 3,072 | 9.4% | +5.5 |
| GQA | indexer | 32k | block | 4,096 | 12.6% | +4.3 |
| GQA | indexer | 32k | block | 8,192 | 25.2% | **+1.1** |
| GQA | indexer | 32k | token | 1,024 | 3.1% | +6.3 |
| GQA | indexer | 32k | token | 2,048 | 6.3% | **+2.0** |
| GQA | indexer | 32k | token | 3,072 | 9.4% | **+0.8** |
| GQA | indexer | 128k | block | 8,192 | 6.4% | +6.0 |
| GQA | indexer | 128k | block | 16,384 | 12.7% | **+3.5** |
| GQA | indexer | 128k | token | 2,048 | 1.6% | +4.0 |
| GQA | indexer | 128k | token | 8,192 | 6.4% | **+3.2** |

compute the mean feature once per tile and score each key once against it (Eq. 3). The lightning indexer of DeepSeek-V3.2, which is built for MLA, has neither property. It gives all heads one list, which suits MLA because all heads read the same keys. It applies the ReLU to each query–key score, so scoring a tile of $q$ queries costs $q$ times as much as scoring one query.

**Capacity.** A linear indexer is limited by its capacity, and one hidden layer brings rank 128 to the level of rank 256 (Table 9). The rank $r$ of the features sets the cost of the indexer, so we look for the smallest indexer whose error is not limited by its capacity. With linear feature maps the gap falls from rank 64 to rank 256. Rank 512, or rank 256 with a hidden layer, does not lower the gap further. We therefore deploy rank 128 with one hidden layer of width 128. Per query token and head, it takes 1.2 to 3.6% of the forward FLOPs of full attention (Table 8).

**Settings.** The indexer of Eq. 3 has, per head, two feature maps of two linear layers each ($d \to 128 \to 128$) with a ReLU after each layer. The query features are scaled by $1/\sqrt{d}$ before the output ReLU. The warm-up trains the indexer at 20× the model's learning rate ($10^{-3}$ on Wan2.2-5B, $2 \times 10^{-3}$ on daVinci-MagiHuman). In the sparse stage the indexer is trained at the model's learning rate by the same Muon optimizer as the model, with the KL term weighted by $10^{-5}$ in the total loss.

Table 6: **Scoring a tile through a summary of its queries** (Wan2.2-5B, 16k, $k = 2{,}048$). The 128 queries of a tile share one list of $k$ keys, and every key is a single token. The scorers differ only in how they summarise the 128 queries before they score each key. One model trained with exact selection is evaluated with each scorer, and the learned indexer is taken from a sparse training run with the same data. "Cost" is the multiply-adds of the scoring pass per (tile, key), relative to exact scoring, without the $k$-means clustering. "Gap" is the validation loss minus that of exact selection on the same model. The mean query is the query summary of VSA (Zhang et al., 2025b), which also averages the keys of each block. Here every key is scored on its own.

| Scorer | Cost | Gap |
|---|---|---|
| Exact scores of all 128 queries (Eq. 2) | 1 | 0 |
| Exact scores of 64 of the 128 queries | $1/2$ | $+0.0020$ |
| Exact scores of 32 of the 128 queries | $1/4$ | $+0.0075$ |
| Exact scores of 16 of the 128 queries | $1/8$ | $+0.0084$ |
| Exact scores of 16 $k$-means centres of the queries | $1/8$ | $+0.0017$ |
| Exact scores of 8 $k$-means centres of the queries | $1/16$ | $+0.0031$ |
| Exact scores of 4 $k$-means centres of the queries | $1/32$ | $+0.0055$ |
| Exact scores of the mean query | $1/128$ | $+0.0180$ |
| Learned indexer, linear, rank 128 | $1/128$ | $+0.0008$ |

Table 7: **One list of keys per head against one list shared by all heads** (Wan2.2-5B, 24 heads, 16k, $k = 2{,}048$, 128-query tile). Keys are ranked by their exact scores (Eq. 2), summed per block for block routing. Validation loss at step 5,000. Full attention reads 0.3470.

| Selection | One list per head | One list for all heads | Difference |
|---|---|---|---|
| Token-level | 0.3469 | 0.3473 | $+0.0004$ |
| Block routing | 0.3475 | 0.3489 | $+0.0014$ |

## D Attention Analysis

**Attention kept by token, block and window selection.** To keep the same share of the attention, whole blocks need about twice as many keys as single tokens (Table 10). We measure this on the released checkpoints of the two backbones, on held-out 32k clips. For a query tile, we compute its exact attention over all keys and how much of it $k$ selected keys keep. We select the $k$ keys in three ways: the single tokens with the highest attention, the whole 128-key blocks with the highest attention, and the keys nearest to the tile in the latent grid. With the same $k$, single tokens keep the most attention and the nearest keys the least. To keep $80\%$ of the attention, blocks need 1.9 times the keys of tokens on Wan2.2-5B and 2.2 times on daVinci-MagiHuman.

## E Kernel Details

The backward pass is where token-level selection departs from existing kernels. FlashAttention's backward (Dao, 2024) is organised around keys: a thread block owns a block of keys and values, sweeps over the queries that attend to them, and accumulates $dK$ and $dV$ in registers. This needs to know, for every key, which queries attend to it. Under dense or block-sparse attention that set is regular, a list of whole query blocks. Under token-level selection it is the transpose of the selection, an irregular list that differs from key to key, changes whenever the selection changes, and is not materialised until the selection is sorted. We therefore organise the backward around queries, mirroring the forward. A thread block takes the rows of a tile, re-gathers the tile's keys and values in chunks, recomputes the attention probabilities from the log-sum-exp saved by the forward instead of storing them, and forms all three gradients from the chunk at hand. $dQ$ belongs to rows the tile owns and accumulates in registers. $dK$ and $dV$ are contributions to keys the tile does not own, and are scattered back to their rows. Two details make the scatter work. A key receives one contribution from every tile that selected it, and summing these in bf16 loses the small ones, so the sum is taken in fp32 and cast once at the end. And a tile's rows contribute to a key together, so their contribution is reduced in registers before it leaves the thread block. The backward then writes one contribution per (tile, key) rather than one per (query, head, key), and the sharing that fills the hardware tile in the forward divides the number of contributions by the same factor $q \cdot h_g$. On Wan2.2-5B the contributions are added with fp32 atomic adds. On daVinci-MagiHuman each tile writes its contributions to a bf16

Table 8: **Forward FLOPs of the learned tile indexer relative to full attention**, per query token and head. Full attention takes $2Nd$ multiply-adds and the indexer $Nr/q + 2(dr + r^2)$, ignoring edge tiles, with $d = r = 128$. $N$ is the number of video tokens and $q$ the number of queries in a tile.

| Backbone | Context | $N$ | $q$ | Indexer / full attention |
|---|---|---|---|---|
| Wan2.2-5B (MHA) | 16k | 17,024 | 128 | 1.9% |
| | 32k | 32,560 | 128 | 1.2% |
| daVinci-MagiHuman (GQA) | 16k | 17,024 | 24 | 3.6% |
| | 32k | 32,560 | 24 | 2.9% |
| | 128k | 128,520 | 24 | 2.3% |

Table 9: **Capacity of the learned tile indexer** (Wan2.2-5B, 16k, $k = 2{,}048$, one selection per head over a 128-query tile). Every indexer is an earlier version of the deployed design, distilled from the first sparse step toward the exact tile scores over all keys. Gap to full attention in units of $10^{-4}$, the mean over the six validation reads from step 1,000 to 2,000 of sparse training.

| Feature maps | Gap |
|---|---|
| Linear, rank 64 | +8.5 |
| Linear, rank 128 | +7.0 |
| Linear, rank 256 | +5.7 |
| Linear, rank 512 | +5.8 |
| Rank 128, one hidden layer of width 128 | +5.5 |
| Rank 256, one hidden layer of width 256 | +5.5 |

buffer, and a second pass sums the contributions of each key in fp32 in a fixed order, which makes the backward deterministic.

We implement the forward and the per-tile backward in CuTe. The forward pass extends the kernel released with Keye-VL-2.0 (Kwai Keye Team, 2026) to head groups that do not divide the 128-row tile. The backward pass is ours and follows the design above. On daVinci-MagiHuman the backward sorts the selection with CUB before the CuTe kernel runs, and its second pass is a Triton kernel. The Wan2.2-5B runs with exact scores (Fig. 3a) execute the same selection with PyTorch's FlexAttention.

## F Speed Measurement Details

**Timing protocol.** A forward time is the median of 10 forwards of the whole denoiser after 2 warm-up forwards, on one clip, or at 128k on successive validation clips, and sampling repeats this forward at every denoising step. A layer time is the median of 20 CUDA-event timings of one sparse-attention layer on random inputs at the token grid, text tail and lossless budget of the variant, on the sampling path, which computes the indexer scores, the top-$k$, the gather and the attention over the selected keys and the text tail. On the 15B backbone the layer holds all 40 query heads and 8 key–value heads at 16k and 32k, and at 128k the five query heads and one key–value head that one GPU holds under 8-way context parallelism. Its reference is the dense attention layer of the full-attention model, the flex-attention kernel of MagiAttention, within 8% of FlashAttention-3 at these shapes. A kernel time runs only the attention calls of that layer, with their wrappers. On Wan2.2-5B this is the sparse-attention wrapper, which also packs the query rows. On the 15B backbone these are the sparse and dense attention calls captured from one forward. The layer time adds the scoring and selection of the keys and the work outside these calls. The top-$k$ of the training code selects the same keys as FlashInfer's kernel and runs slower, and with it the forward speed-up is 1.43× on the 5B backbone at 32k and 1.90× on the 15B backbone at 128k.

**Layer against kernel, VTSA against block routing.** The layer adds the indexer scores, the selection and the work outside the attention calls to the kernel. The scores are the same for VTSA and block routing and move the ratio toward one. The selection of block routing sums a softmax over all keys per block before it ranks the blocks, costs more than the top-$k$ of VTSA and moves the ratio the other way. At the same $k$ the scattered keys of VTSA take 1.46 ms against 1.40 ms for whole 128-token tiles at 16k and 3.31 against 3.25 ms at 32k on Wan2.2-5B. The layer ratio against block routing lands 0.1–0.2 below the kernel ratio on Wan2.2-5B and 0.1–0.4 below it on the 15B backbone. On the 15B backbone the forward with block routing at its lossless budget takes 1,467 ms

Table 10: **Attention kept by a selection of $k$ keys at 32k tokens.** Share of a query tile's exact attention weight (averaged over its queries and heads, summing to one over the keys) kept by the exact top-$k$ tokens, by the highest-scoring whole blocks (up to 128 tokens, the last one truncated) with the same number of keys, and by the $k$ nearest keys in the latent grid, and the number of keys at which the mean share reaches 80%, as tokens and as blocks, interpolated in $\log_2 k$ between the two tested budgets around it. Wan2.2-5B: 32 held-out clips at timesteps 860, 600 and 300, 16 query tiles in each of the 30 layers. daVinci-MagiHuman: 24 held-out clips at their training timesteps, in the 39 layers that use sparse attention.

| | | Kept at $k$ | | | Keys to keep 80% | | |
|---|---|---|---|---|---|---|---|
| Checkpoint | $k$ ($\rho$) | Tokens | Blocks | Nearest | Tokens | Blocks | Ratio |
| Wan2.2-5B, released | 3,072 (9.4%) | 0.85 | 0.78 | 0.61 | 1,871 | 3,607 | 1.9 |
| daVinci-MagiHuman, released | 2,048 (6.3%) | 0.74 | 0.59 | 0.37 | 3,415 | 7,515 | 2.2 |

against 1,406 ms with full attention at 16k and is 1.5× faster at 128k. At 30% of the keys the block kernel takes half of the dense layer (5.0 against 9.7 ms at 16k), and the indexer scores, the selection and the work outside the attention calls take about as much again (6.9 ms), so the block layer (11.9 ms) is slower than the dense layer. VTSA spends 3.1 ms in the kernel and 4.9 ms outside it. The same holds at 32k at 25% of the keys.

## G Validation Resolution

A gap between two models means something only if it is larger than the differences that the evaluation produces when the model does not change. We measure these differences in three ways. (1) Clip sampling. Every validation set holds 1,280 clips, each with a fixed noise and timestep. We compare two models by their loss difference on each clip, averaged over the clips. This average has a 95% confidence interval of $\pm 1.96\, s/\sqrt{1{,}280}$, where $s$ is the standard deviation of the per-clip differences, which is $\pm 3.1\times10^{-4}$ on the MHA backbone at 16k. This is how much the gap can move because of which clips were drawn. (2) Training seed. We train full attention twice on the MHA backbone at 16k, changing only the random seed. The difference between the two validation losses has a standard deviation of $1.9\times10^{-4}$ and is never larger than $5\times10^{-4}$. (3) Replay. We resume a full-attention run from a checkpoint and train the same steps again. At every validation read of the repeated steps, the loss matches the first pass within $3\times10^{-4}$. Averaging the ten reads of the read window does not remove the first source, because every read uses the same clips. All three measurements lie at 2–$5\times10^{-4}$, so we set the resolution of the evaluation (§3.1) at $4\times10^{-4}$, and a gap below it is of the size of the difference between full attention and itself. This threshold is the radius of a region of practical equivalence around full attention (Kruschke, 2018), and a smallest effect size of interest (Lakens, 2017; Lakens et al., 2018). Bouthillier et al. (2021) make the same point for benchmark scores, that a difference between methods has to be read against the variation of the evaluation itself.

## H Generation Quality Details

**Prompts and sampling.** Every prompt is rewritten by Qwen2.5-7B-Instruct before generation, with the system prompt released with Wan2.2 on Wan2.2-5B and the prompt guide released with daVinci-MagiHuman on daVinci-MagiHuman. Every model generates at 704×1,280 pixels and 145 frames, the training geometry, with the same sampler for every model of a backbone: 50 sampling steps, guidance 5 and shift 5 on Wan2.2-5B, and 32 steps, guidance 5 (lowered to 2 for timesteps at or below 500) and shift 5 on daVinci-MagiHuman. On daVinci-MagiHuman the difference between the conditional and unconditional video predictions has its mean over the clip removed in each channel before the guidance scale is applied. On both backbones each trained model is the average of its checkpoints at 3,000, 3,500, 4,000, 4,500 and 5,000 updates.

**VBench-2.0.** The human dimensions of VBench-2.0 (Zheng et al., 2025) are scored on their own prompts, rewritten as above, with three clips per prompt from three initial noises shared by all models; the second full-attention row generates from three further initial noises. For each clip,

Human Anatomy is the share of the people in its frames that the benchmark's classifiers judge free of anatomical defects, Human Clothes is whether its video language model judges the clothing consistent through the clip, and Human Identity is the share of frames whose face matches the face in the first frame by face-recognition similarity. Each dimension averages its clip scores, and Human Fidelity is the unweighted mean of the three dimensions, as on the benchmark's leaderboard.

# I GENERATION GALLERY

Figs. 5 and 6 show three prompts on Wan2.2-5B and two on daVinci-MagiHuman at the equal budget of Fig. 4, where block routing and VTSA both select $k = 3{,}072$ keys ($\rho = 9.4\%$). Entries W1 and M1 are the prompts of Fig. 4.

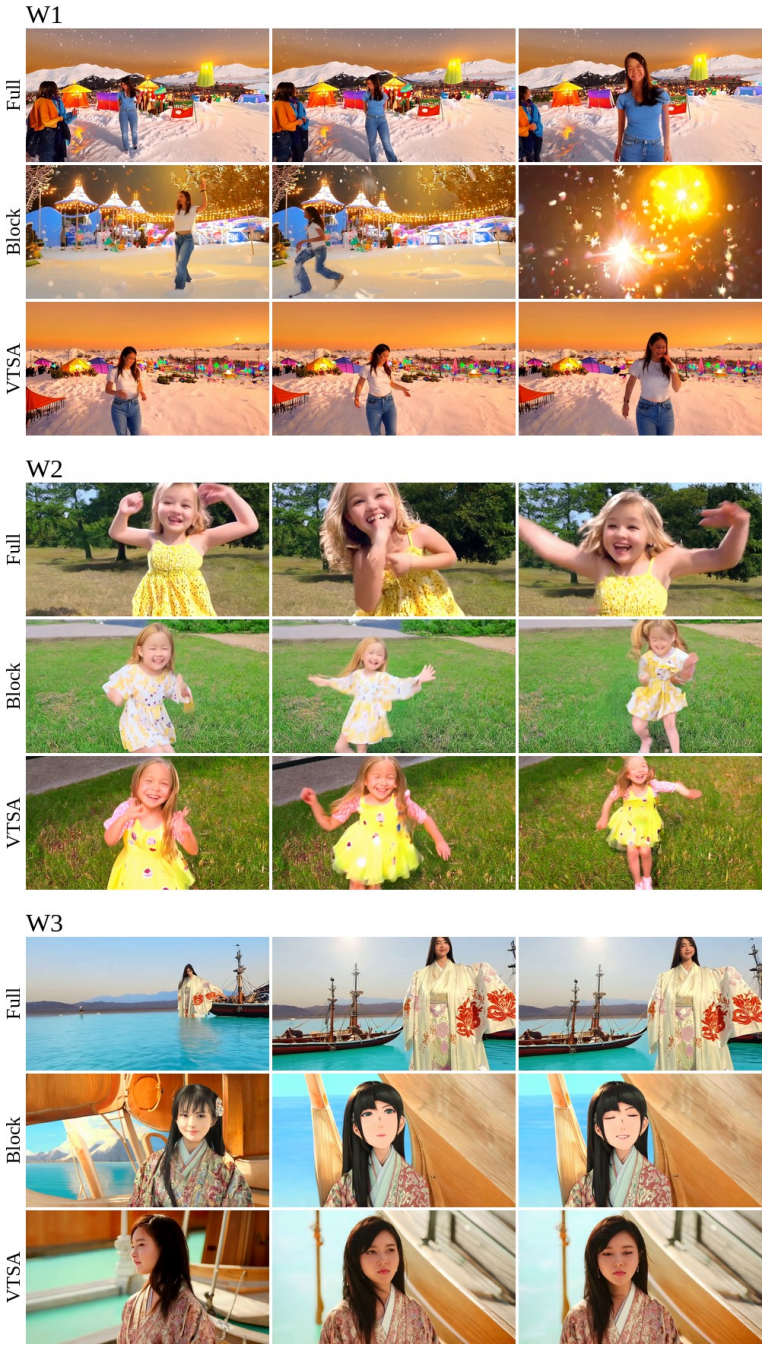



Figure 5: **Further videos generated at 32k on Wan2.2-5B by full attention, and by block routing and VTSA at an equal budget.** Rows of each entry: full attention, block routing at $k = 3{,}072$ and VTSA at $k = 3{,}072$, on the same prompt. Prompts (abridged): W2 "a young girl jumps up and down with joy" and W3 "a young Japanese woman in a kimono stands on the deck of a sailing ship".

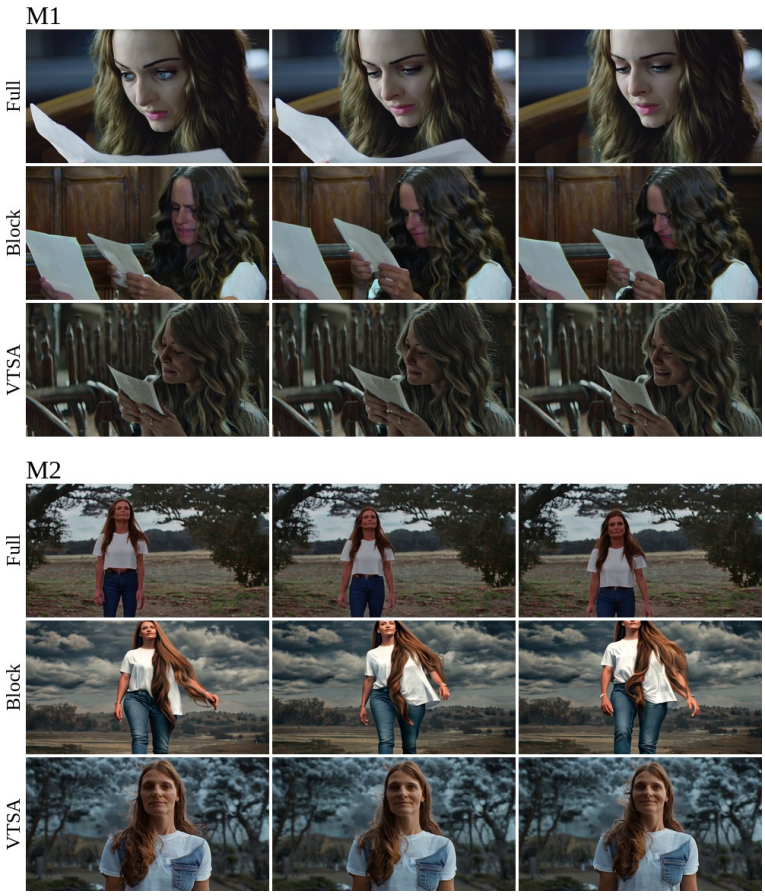



Figure 6: **Further videos generated at 32k on daVinci-MagiHuman by full attention, and by block routing and VTSA at an equal budget.** Rows of each entry: full attention, block routing at $k = 3{,}072$ and VTSA at $k = 3{,}072$, on the same prompt. Prompt of M2 (abridged): "a woman in jeans and a white T-shirt stands still".