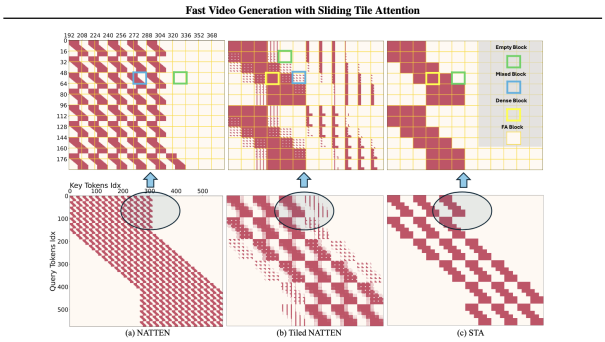
Sliding tile attention

A Preprint

Paper is based on 3D attention which is used in DiT. The main bottleneck is scaling because as video size scales we got attention scaling quadratic. Video data exhibit high redundancy – adjacent frames exhibit minimal differences. Authors suggest that window that covers only 15% of tokens accounts for 70% of total attention which is crazy. Tile - a contiguous group of tokens forming a spatial-temporal cube, with its size determined by the block size in FlashAttention. In SWA, a query attends only to keys within a fixed window, and stacking multiple attention layers naturally expands the receptive field beyond the window size(The reason that stacking multiple attention layers naturally expands the receptive field beyond the window size is because each layer's output is based on the attended information from the previous layer within its local window. When these layers are stacked, the "local window"effectively expands).

Problem with inefficiency: attention blocks into three types: dense (with all attention scores retained), empty (mask out all values), and mixed (with some scores removed). While empty can be literally skipped mixed introduce a significant overhead. irst, they require full computation for the entire block before applying masks to retain or discard attention scores, introducing unnecessary computations. Second, to determine which position to retain or discard, the attention kernel needs to calculate the value of the mask based on the SWA pattern and the block's position relative to the entire attention mask.



Overall method operates only on dense or empty blocks.

Certainly, here are those theorems formatted in LaTeX:

Theorem 3.1. Consider a tiled NATTEN configuration with tile size (T, T, T), window size (W, W, W), and video size (L, L, L). Let the FA block size be (B, B), where $B = T^3$. Ignoring boundary effects, the number of dense blocks is given by:

$$N_{\text{dense}} = \left(\max\left(2\left\lfloor\frac{W+1}{2T}\right\rfloor - 1, 0\right)\right)^3 \cdot \left(\frac{L}{T}\right)^3$$

The number of mixed blocks in tiled NATTEN is:

$$N_{\text{mix}} = \left(2\left\lfloor\frac{W-1}{2T}\right\rfloor + 1\right)^3 \cdot \left(\frac{L}{T}\right)^3 - N_{\text{dense}}$$

Theorem 3.2. With the same notation, if W is an integer multiple of T, the number of dense blocks in SLIDING TILE ATTENTION (STA) is:

$$S_{\text{dense}} = \left(\frac{W}{T}\right)^3 \cdot \left(\frac{L}{T}\right)^3,$$

and all remaining blocks are empty. There are no mixed blocks in STA. :contentReference[oaicite:5]index=5

Implementation notes

STA flattens tokens so that tokens within a tile have consecutive indices. STA slides by tile steps (T, T, T). Producer warpgroups decide which KV tile blocks to load; consumer warpgroups compute dense attention on SRAM-resident query blocks. This hides mask logic in the producer side and leaves compute warpgroups oblivious to sparsity. Implementation builds on FlexAttention + ThunderKittens / FA3 ideas.

Kernel-level optimizations used by authors:

- split threadblock into compute warpgroups and data warpgroups;
- asynchronous KV loading from HBM to SRAM for only needed key tiles;
- disaggregate inter-block mask logic from compute kernel (producer handles mask).

These choices enable skipping empty blocks and avoid intra-block masks on dense blocks. :contentReference[oaicite:7]index=7

Key results (short)

- STA removes mixed blocks entirely and therefore translates sparsity into real wall-clock speed. :contentReference[oaicite:8]index=8
- STA MFU (optimized kernel) = 58.79%. End-to-end on HunyuanVideo: FA3 945s → STA 501s (no training). Fine-tuning further reduces to 268s with tiny quality drop. STA gives up to ∼3.53× end-to-end speedup vs FA3.
- Kernel speedups: STA achieves up to 10.45× speedup over full attention in ThunderKittens setup; STA keeps kernel efficiency high vs other SWA methods. See table of kernel numbers.