XAttention: Unlocking the Power of Block Sparse Attention with Antidiagonal Scoring

Ruyi Xu*, Guangxuan Xiao*, Haofeng Huang, Junxian Guo, Song Han

Tsinghua University
Massachusetts Institute of Technology
SJTU
NVIDIA

***: Equal Contribution**

Motivation

Deploying Long-Context LLMs is Crucial But Challenging

- LLMs need to handle long-context like summarizing long texts and processing images/videos.
- Prefilling Memory and Latency Increase Quadratically with Context Length
- Sparse attention can be used to address this issue.

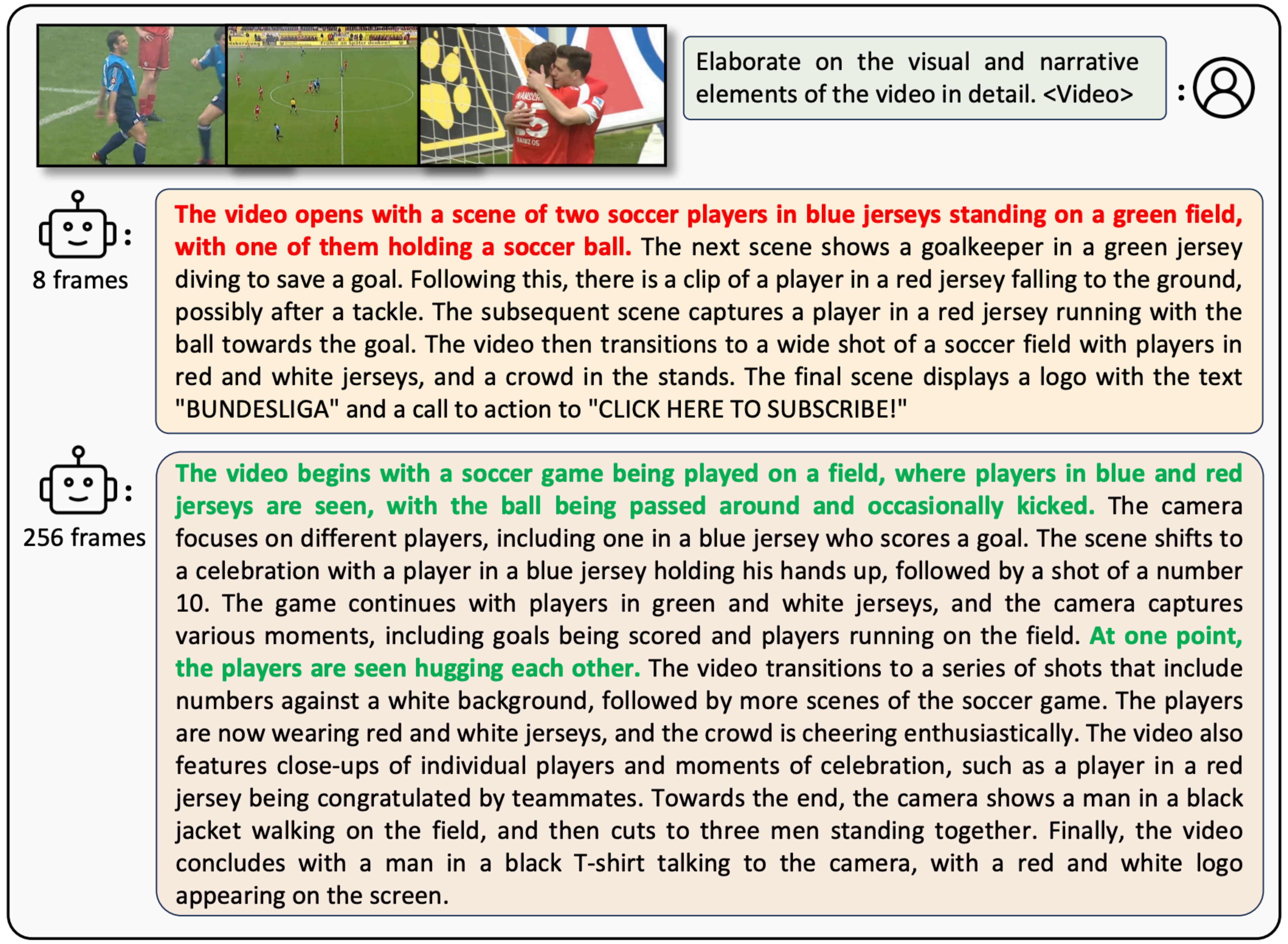


Table 5. Density on Different Context Lengths. Stride $S = 8$ achieves lower sparsity, and as context length increases, sparsity generally increases (lower density).

SeqLen	Stride 4	Stride 8	Stride 16
4k	51.73%	52.16%	55.38%
8k	40.96%	43.77%	43.55%
16k	27.43%	27.49%	28.91%
32k	21.09%	20.97%	27.93%
64k	9.43%	10.98%	11.32%
128k	6.20%	6.89%	7.32%

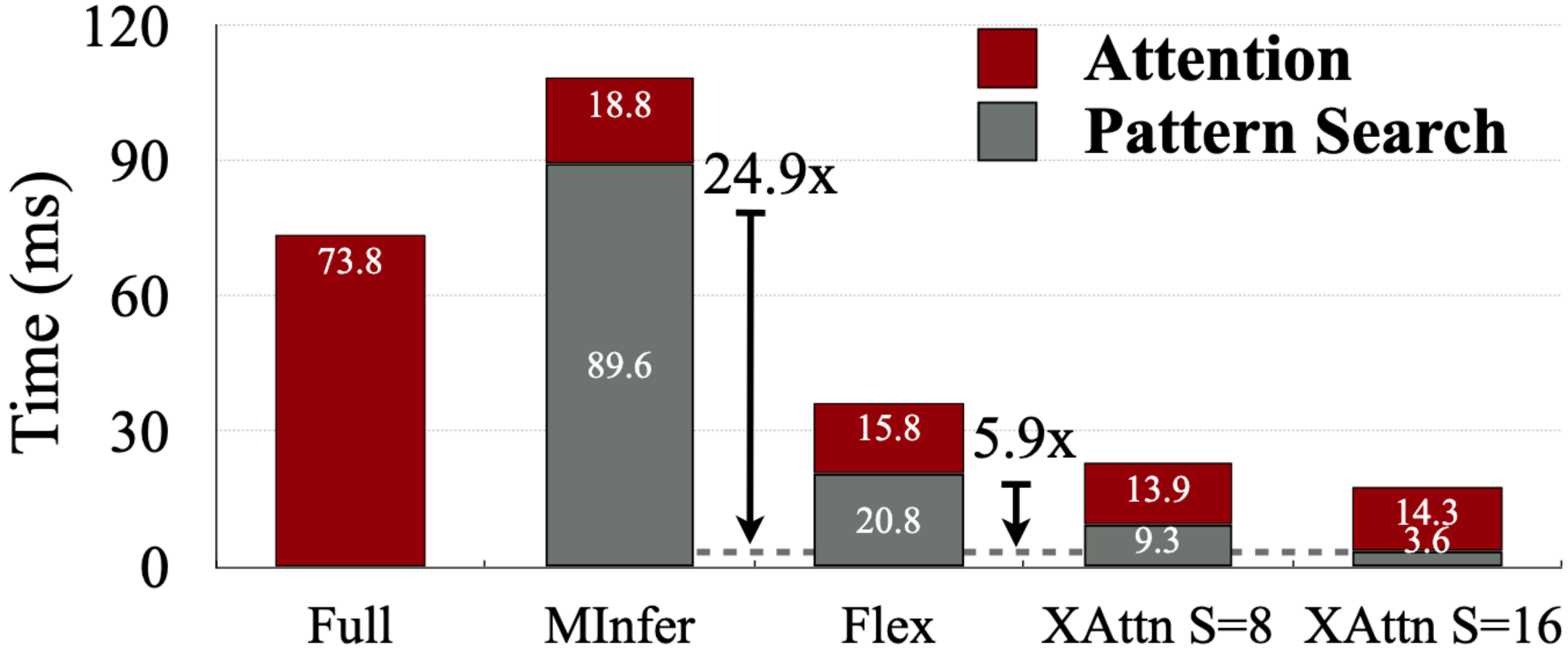
a 224×224 image = 256 tokens

a 1-hour video at 1 FPS = **1 million tokens**

Key Challenges

Block Pooling, Index Search and Accuracy

- **Block Pooling:** Current methods use block pooling to predict the importance of attention blocks
- **Index Search:** To achieve lossless accuracy, index search is required, which is **time-consuming**.
- Prediction method should automatically and robustly identify significant patterns, including crucial vertical and slash patterns.



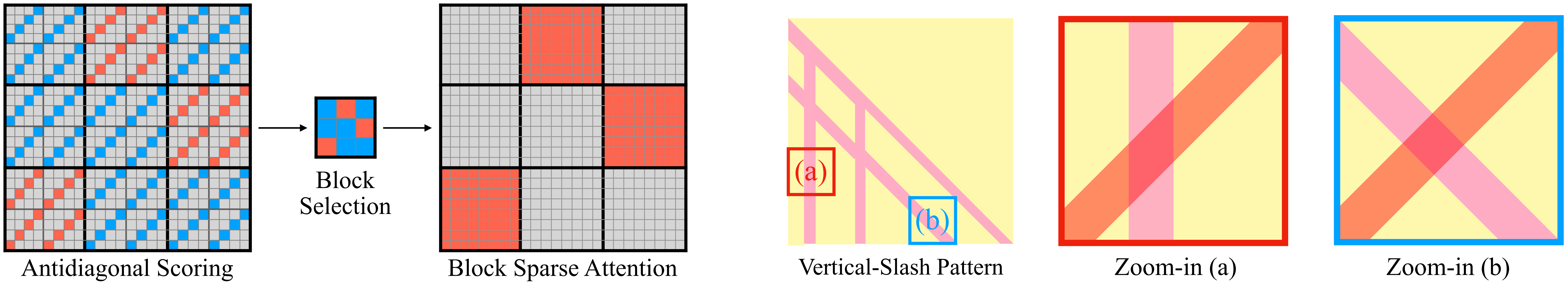
XAttention: Unlocking the Power of Block Sparse Attention with Antidiagonal Scoring

Importance Prediction

Antidiagonal Selection Method

- Within each block of size B, we select elements along the antidiagonal using a stride S to predict importance of the whole block.

- **Information Preservation:** Ensure that information from all tokens is considered, as each token contributes to at least one antidiagonal sum.
- **Pattern Detection:** Antidiagonal intersects every possible vertical and slash pattern within a block



XAttention: Unlocking the Power of Block Sparse Attention with Antidiagonal Scoring

Threshold Block Selection

Dynamically determine density according to context.

- **Antidiagonal sum:** Select elements along the antidiagonal within each S × S block of the attention map and compute the sum of these elements for each antidiagonal.
- **Softmax normalization:** Apply the softmax function to these antidiagonal sums, yielding a probability distribution
- **Block selection:** identify the minimal set of blocks whose cumulative sum of antidiagonal probabilities exceeds a predefined threshold τ .

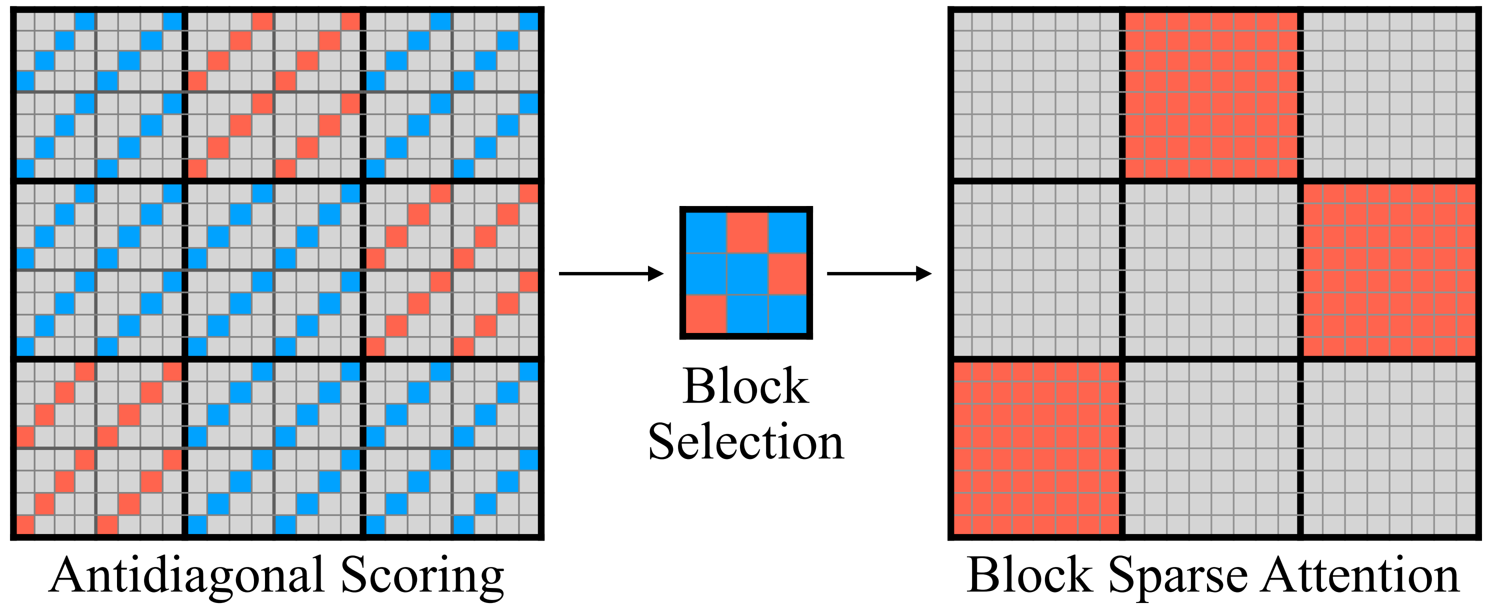
$$\texttt{find\_blocks}(A, \tau) = \arg\min_{\mathcal{B}} \left\{ |\mathcal{B}| \;\middle|\; \sum_{b \in \mathcal{B}} \sum_{(i,j) \in b} A_{i,j} \geq \tau \right\}$$

Algorithm 1 Block Selection

Require: Query matrix $Q \in \mathbb{R}^{L \times d}$, Key matrix $K \in \mathbb{R}^{L \times d}$, block size B, stride S, head dimension d_h, threshold τ

Ensure: Sparse mask M

$N_B \leftarrow \lfloor L/B \rfloor$ {Number of blocks}
for $b = 0$ to $N_B - 1$ **do**
  $Q_{\text{slice}} \leftarrow Q[bB : (b+1)B, :]$ {Extract Q block}
  $Q_{\text{reshaped}} \leftarrow []$
  for $i = S - 1$ down to 0 **do**
    $Q_{\text{reshaped}}.\text{append}(Q_{\text{slice}}[i :: S, :])$ {Reshape along antidiagonals with stride S}
  end for
  $K_{\text{reshaped}} \leftarrow []$
  for $i = 0$ to $S - 1$ **do**
    $K_{\text{reshaped}}.\text{append}(K[i :: S, :])$ {Reshape along antidiagonals with stride S}
  end for
  $A_{\text{approx}} \leftarrow \text{Softmax}\left(\frac{Q_{\text{reshaped}} K_{\text{reshaped}}^T}{\sqrt{d_h} \cdot S}\right)$ {Approximate attention scores}
  $M_b \leftarrow \text{find\_blocks}(A_{\text{approx}}, \tau)$ {Find blocks based on threshold}
end for
$M \leftarrow \text{concatenate}(M_0, M_1, \ldots, M_{N_B - 1})$ {Concatenate block masks}



Minimum Threshold Prediction

Dynamic programming to determine the optimal threshold for each attention head.

- **Problem Formulation:** Consider a model with H attention heads.
- We define a dynamic programming table D[h][m], where h ∊ {1, 2, ..., H} represents the h-th head, and m ∊ {1, 2, ..., M} denotes the number of threshold adjustments made.

$$D[h][m] = \max(D[h-1][m], P(h, m))$$

- **Dynamic Programming：** D[h][m] stores the best performance achievable when exactly m threshold adjustments have been made across the first h heads.
- $t_h(m) = t_h(m-1) \times 0.9$
- This Further reduces the density and computational cost of Xattention.

Stride	$S=4$		$S=8$		$S=16$	
Metric	Avg	Density	Avg	Density	Avg	Density
$\tau = 0.9$	87.51	23.06%	84.96	26.13%	85.83	28.36%
Minimum τ	**88.89**	**21.09%**	**88.47**	**20.97%**	**88.08**	**27.93%**

Results on Accuracy Benchmarks

Long-context Benchmarks: RULER and LongBench

- **RULER:**

Input Len	4k	8k	16k	32k	64k	128k	Avg.
Full	96.74	94.03	92.02	84.17	81.32	76.89	87.52
FlexPrefill	95.99	93.67	92.73	88.14	81.14	**74.67**	87.72
MInference	96.54	94.06	91.37	85.79	83.03	54.12	84.15
SeerAttn	84.43	79.55	79.80	72.95	64.79	51.61	72.18
Xattn S=8	**96.83**	**94.07**	93.17	**90.75**	**84.08**	72.31	**88.47**
Xattn S=16	96.11	93.95	**93.56**	90.64	83.12	71.11	88.08

- **LongBench:**

	Single-Doc QA			Multi-Doc QA			Summarization				Few-shot Learning			Code			
Method	NrtvQA	Qasper	MF-en	HPQA	2WikiMQA	MuSiQue	GovReport	QMSum	VCSum	MultiNews	TREC	TriviaQA	SAMSum	LSHT	LCC	RB-P	Avg.
Full	31.44	25.07	29.40	16.89	17.00	11.79	34.22	23.25	15.91	26.69	72.50	91.65	43.74	46.00	52.19	49.14	40.34
MInference	**31.59**	24.82	**29.53**	17.03	**16.46**	11.58	34.19	23.06	16.08	26.71	**72.50**	**91.18**	43.55	46.00	52.33	49.93	40.30
FlexPrefill	27.30	**28.56**	27.66	17.20	15.14	9.46	32.76	**23.66**	16.05	**27.25**	64.00	88.18	41.28	31.00	45.69	47.54	36.83
XAttention	30.48	26.04	29.28	**17.33**	16.34	**11.88**	**34.60**	23.24	**16.11**	27.08	71.50	90.97	**44.13**	**46.50**	**53.23**	**50.94**	**40.54**

Results on Accuracy Benchmarks

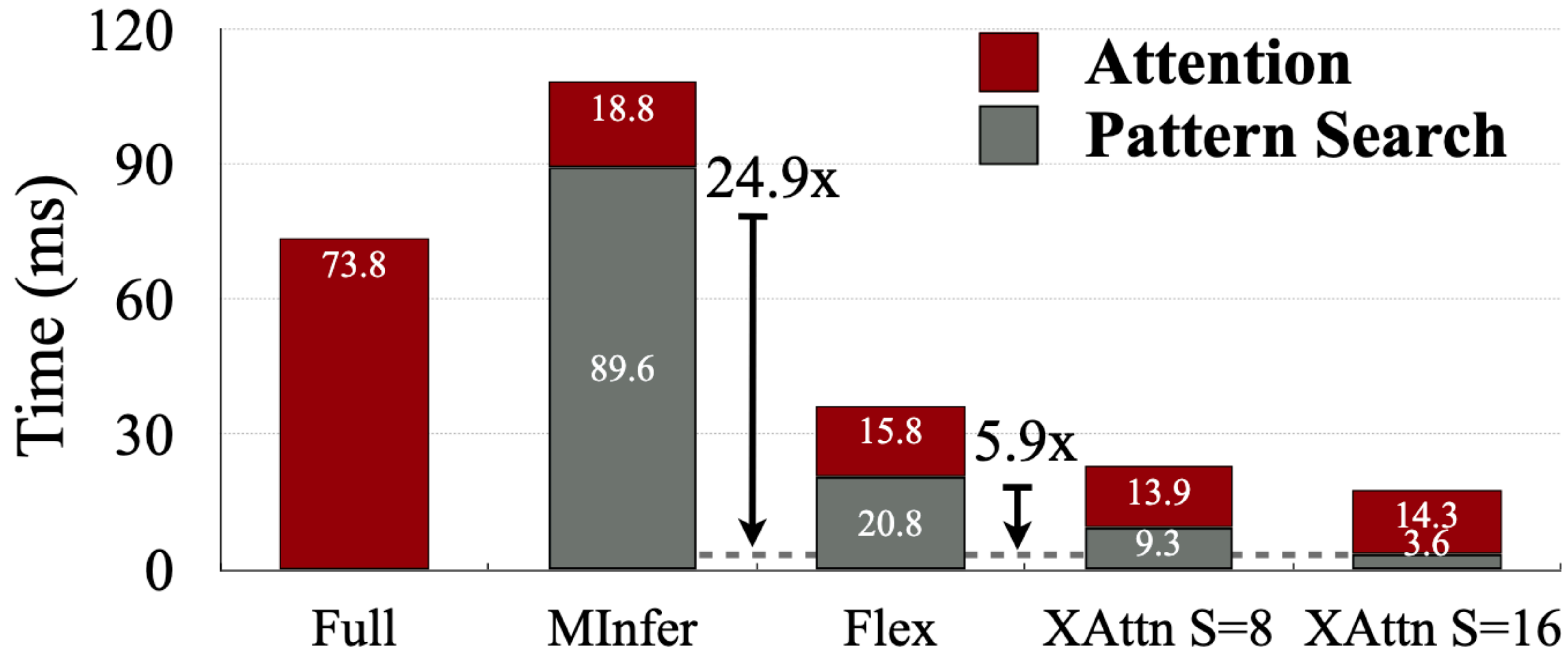
Video Understanding Benchmark: Video-MME

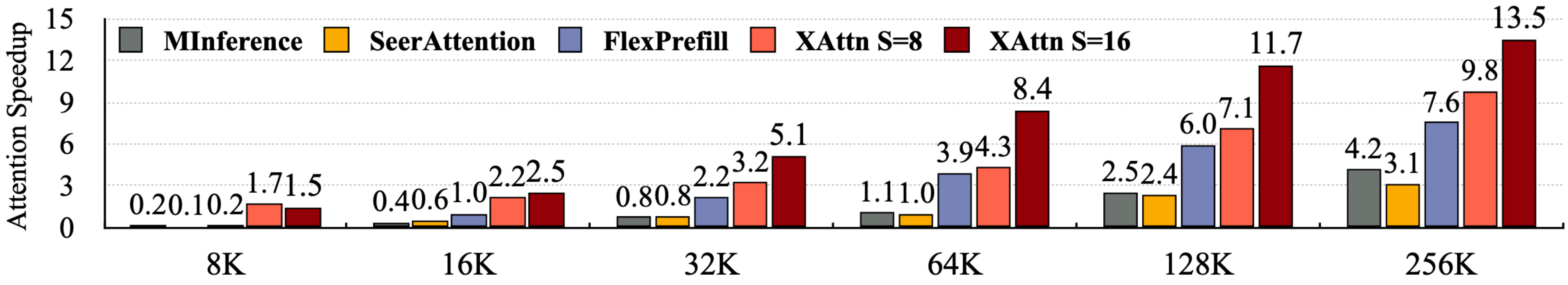
- Xattention demonstrates good transferability on the QwenVL-2-7B model.
- XAttention outperforms Full Attention on long video tasks and achieves the best average performance among all sparse attention methods.

	Short (%)		**Medium (%)**		**Long (%)**		**Overall (%)**	
subs	w/o	w/	w/o	w/	w/o	w/	w/o	w/
Full	72.1	78.1	63.9	69.4	55.1	60.2	63.7	69.2
MInference	71.7	77.6	62.3	67.9	55.2	59.8	63.1	68.4
FlexPrefill	71.4	77.4	**62.6**	68.3	53.8	57.3	62.6	67.7
XAttention	**71.9**	**78.8**	**62.6**	**68.5**	**55.7**	**60.3**	**63.3**	**69.1**

Prefilling Latency Improvements

- **XAttention provides up to 13.5× decoding latency improvement for Llama-3-8B-Instruct model**
- **XAttention accelerates pattern search time by up to 24.9x compared to Minference and 5.9x compared to Flexprefill.**





Ablation Study

- **Antidiagonal Pattern**

	Stride $S = 8$			Stride $S = 16$		
Metric	32k	Avg.	Density	32k	Avg.	Density
Random	82.53	82.48	27.57%	82.35	80.94	31.36%
Diagonal	76.47	81.06	24.47%	58.26	79.63	25.31%
Antidiagonal	**90.75**	**88.47**	20.97%	**90.64**	**88.08**	27.93%

- **Top-K vs. Top-Ratio vs. Dynamic**

Stride	$S = 4$		$S = 8$		$S = 16$	
Metric	Avg	Density	Avg	Density	Avg	Density
Top K	84.96	17.40%	84.13	19.92%	83.11	30.15%
Ratio	85.96	21.00%	85.42	21.00%	84.24	27.00%
Threshold	**88.89**	21.09%	**88.47**	20.97%	**88.08**	27.93%

- **Minimum Threshold Prediction**

Stride	$S = 4$		$S = 8$		$S = 16$	
Metric	Avg	Density	Avg	Density	Avg	Density
$\tau = 0.9$	87.51	23.06%	84.96	26.13%	85.83	28.36%
Minimum τ	**88.89**	**21.09%**	**88.47**	**20.97%**	**88.08**	**27.93%**

- **Stride Sizes**

Stride	$S = 4$	$S = 8$	$S = 16$	$S = 64$
Avg	88.89	88.47	88.08	81.21
Density	21.09%	20.97%	27.93%	39.88%

Conclusion

- **We present XAttention, a novel plug-and-play framework for accelerating long-context inference in Transformer models**
- **Code: https://github.com/mit-han-lab/x-attention**