AdaSpark: Adaptive Sparsity for Efficient Long-Video Understanding

Supplementary Material

1. Effect of Selection Strategy

Adhering to the ablation experimental setup described in the main text, we investigate the influence of various token selection strategies under an identical compression ratio. As summarized in Table 1, we initially evaluate the most rudimentary approach: uniform sampling. This method exhibits the most significant performance degradation under equivalent compression levels. Subsequently, we report the performance of the static Top-K strategy, which serves as our primary comparative baseline. Due to its inability to dynamically select visual cubes across different layers, the Top-K approach lags behind our method, resulting in performance deficits of 1.9 on Charades STA, 3.1 on VideoMME, and 2.4 on MLVU. We further explore a constrained variant that applies Top-K selection exclusively to fixed I-frames (keyframes identified by traditional video compression algorithms); the results indicate that the performance deviation from the standard Top-K approach is negligible. In contrast, our AdaSpark employs a dynamic Top-P mechanism, which facilitates a more flexible selection strategy and yields superior performance.

Table 1. **Ablation on more selection strategy.**

Selection Strategy	Charades STA	VideoMME	MLVU
Uniform Sampling	31.5	50.2	53.8
Top-K	34.2	52.6	55.1
I-Frame	34.0	52.9	55.7
Top-P	**36.1**	**55.7**	**57.5**

2. Implementation Details

2.1. Training Configuration

Table 2 provides a comprehensive summary of the hyperparameters employed in the training of AdaSpark. Throughout this process, the visual encoder is maintained in a frozen state, and we implement a cube-based sparse strategy regulated by a top-p threshold.

Our training methodology incorporates a mixed dataset, centrally featuring the llava-video-178k [10] dataset, which serves as a foundational corpus for basic video understanding. In addition, we augment this data with 77k timestamp-grounded samples from DideMo [1] and ActivityNet Captions [4] to enhance the model's capacity for identifying key temporal information.

2.2. Pseudocode

To provide a comprehensive understanding of our method, we present the pseudocode detailing the AdaS-Attn and AdaS-FFN mechanisms in Algorithm 1.

We provide the implementation of critical components for reference; please refer to `AdaS-Attn.py` and `AdaS-FFN.py` in zip file for specific details.

Table 2. **Hyperparameter configuration for AdaSpark training.** The model undergoes a single-stage post-training protocol.

Hyperparameter	**Value**
Model Configuration	
Backbone	Qwen2.5-VL-3B / 7B
Visual Encoder Status	Frozen
Max Visual Sequence Length	48k
Max Context Length	64k
Cube Size ($H \times W \times T$)	$8 \times 8 \times 4$
Top-p Threshold	0.7
Training Optimization	
Data Scale	$\sim$255K (178K + 77K)
Global Batch Size	256
Learning Rate (lr)	2×10^{-6}
Sequence Parallel	4
LR Schedule	Cosine Decay
Optimizer	AdamW
Weight Decay	0
DeepSpeed Stage	Zero2
Data & Hardware	
Input FPS	4
Compute Resources	32 $\times$ NVIDIA H100
Training Time	$\approx$ 4 Days

Algorithm 1: AdaSpark Training Algorithm

Input: Video Tokens X_{vid}, Text X_{txt}, Cube (t, h, w), p
for *each transformer layer* **do**
  $Q, K, V \leftarrow \text{Linear}(X)$; Apply RoPE$(Q, K)$
  AdaS-Attn: Reshape K, V into Cubes $(B, N, t \cdot h \cdot w, D)$
  For each query, select Top-p relevant cubes based on proxy scores
  Calculate Attention on {Selected Cubes $\cup$ Local Cube};
  AdaS-FFN: Reshape video tokens into Cubes
  Calculate token importance via L_2-norm; Select Top-p salient tokens
  Apply FFN to selected vision tokens and full text tokens; Add Mean(FFN($Active$)) to others
  Restore original sequence shape and add residual connections
end
return Updated X_{seq}

To facilitate a more detailed understanding of the compression achieved by our algorithm on standard causal attention and FFN layers, we provide a comprehensive discussion on the theoretical FLOPs calculation in the following section.

2.3. Detail of FLOPs Calculation

FLOPs Analysis for AdaS-Attn. We compare the theoretical FLOPs of standard dense causal attention with our proposed AdaS-Attn. Let S, D_{model}, and C denote the sequence length, model dimension, and the number of tokens per spatiotemporal cube, respectively. For standard **dense causal attention**, computational costs arise from the Query-Key (QK^T) and Attention-Value (AV) multiplications. Since each query attends to all preceding keys/values (averaging $S/2$ tokens due to the causal mask), the total complexity exhibits a quadratic dependence on sequence length:

$$\begin{aligned} FLOPs_{Dense} &\approx 2 \times S \times \frac{S}{2} \times D_{model} \\ &+ 2 \times S \times \frac{S}{2} \times D_{model} \\ &\approx 2S^2 D_{model} \end{aligned} \tag{1}$$

In contrast, **AdaS-Attn** computes attention sparsely. Let $\bar{N}$ be the average number of top-p cubes selected per query. The attention mechanism is restricted to the tokens within these selected cubes (totaling $\bar{N} \times C$ tokens). The complexity for the sparse QK^T and AV phases is calculated as:

$$\begin{aligned} FLOPs_{AdaS} &= FLOPs_{QK^T} + FLOPs_{AV} \\ &\approx (2S \cdot \bar{N}C \cdot D_{model}) \\ &+ (2S \cdot \bar{N}C \cdot D_{model}) \\ &\approx \mathbf{4}S\bar{N}CD_{model} \end{aligned} \tag{2}$$

Since the number of selected tokens is significantly smaller than the full sequence ($\bar{N}C \ll S$), AdaS-Attn achieves linear complexity $O(S)$, drastically reducing computational overhead compared to the $O(S^2)$ standard attention.

FLOPs Analysis for AdaS-FFN. We further analyze the efficiency gains in the Feed-Forward Network (FFN). Let D_{ff} denote the intermediate dimension of the FFN (typically $4D_{model}$ or similar).In a Standard FFN, every token in the sequence S undergoes projection up to D_{ff} and down to D_{model}. The total FLOPs are dominated by these dense matrix multiplications:

$$\begin{aligned} FLOPs_{Std-FFN} &\approx 2 \times S \times D_{model} \times D_{ff} \\ &+ 2 \times S \times D_{ff} \times D_{model} \\ &= 4SD_{model}D_{ff} \end{aligned} \tag{3}$$

For our AdaS-FFN, computation is content-aware. The additional overhead for calculating L_2-norms and the importance distribution S_i is $O(S \cdot D_{model})$, which is negligible compared to the matrix transformations ($D_{model} \times D_{ff}$).The heavy FFN computation is only applied to the set of activated tokens M_i. Let $S_{act} = \sum |M_i|$ be the total number of activated tokens across all cubes, and $\bar{r} = S_{act}/S$ be the average activation ratio. The inactive tokens R_i bypass the FFN and utilize the Mean Compensation strategy, which involves only lightweight vector addition operations ($O(S \cdot D_{model})$).Thus, the FLOPs for AdaS-FFN are proportional only to the activated tokens:

$$\begin{aligned} FLOPs_{AdaS-FFN} &\approx 4 \times S_{act} \times D_{model} \times D_{ff} \\ &= \mathbf{4} \times (\bar{r} \cdot S) \times D_{model} \times D_{ff} \end{aligned} \tag{4}$$

Given that $\bar{r}$ is controlled by the top-p threshold (typically $\bar{r} \ll 1$), AdaS-FFN significantly reduces the FLOPs by a factor of $\bar{r}$ compared to the standard FFN, while preserving semantic integrity through mean compensation.

3. Evaluation Settings

We evaluate AdaSpark on a series of comprehensive video-language benchmarks: 1) Extra Long Video Understanding, using Video Needle in a Haystack [9, 11]; 2) Long Video Understanding, which includes MLVU [12], VideoMME [2], LongVideoBench [7], and LVBench [6]; 3) Short Video Understanding, using MVBench [5]; 4) Spatial Reasoning, with VSIBench [8]; and 5) Video Grounding, utilizing CharadesSTA [3]. Table 3 details the frame sampling configurations employed during inference via the `lmms-eval` framework. For all tasks, we strictly adhere to the default prompts and scoring protocols provided by the evaluation framework.

Table 3. **Evaluation settings summary for each benchmark.** For all benchmarks, we set the temperature, top p, and number of beams to 0, 0, and 1, respectively. **FPS** denotes the sampling frames per second, and **# F** represents the maximum number of sampling frames allowed.

Benchmark	FPS	# F (Max Frames)
VideoNIAH	1	4096
MLVU (Dev)	1	512
VideoMME	1	256
LongVideo	1	1024
LVBench	1	1024
MVBench	2	256
VsiBench	2	256
CharadesSTA	4	256

References

[1] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*, pages 5803–5812, 2017. 1

[2] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024. 2

[3] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pages 5267–5275, 2017. 2

[4] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715, 2017. 1

[5] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. *arXiv preprint arXiv:2311.17005*, 2023. 2

[6] Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, et al. Lvbench: An extreme long video understanding benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22958–22967, 2025. 2

[7] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024. 2

[8] Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643, 2025. 2

[9] Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024. 2

[10] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024. 1

[11] Zijia Zhao, Haoyu Lu, Yuqi Huo, Yifan Du, Tongtian Yue, Longteng Guo, Bingning Wang, Jing Liu, et al. Needle in a video haystack: A scalable synthetic evaluator for video mllms. In *The Thirteenth International Conference on Learning Representations*. 2

[12] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024. 2