Sparse-to-Dense: A Free Lunch for Lossless Acceleration of Video Understanding in LLMs

Xuan Zhang[1], **Cunxiao Du**[2*], **Sicheng Yu**[1], **Jiawei Wu**[3],
Fengzhuo Zhang[3], **Wei Gao**[1], **Qian Liu**[2]
[1]Singapore Management University, [2]Sea AI Lab, [3]National University of Singapore

Abstract

Video Large Language Models (Video-LLMs) suffer from high inference latency in long video processing due to their auto-regressive decoding mechanism, posing challenges for the efficient processing of video sequences that are usually very long. We observe that attention scores in Video-LLMs during decoding exhibit pronounced sparsity, with computational focus concentrated on a small subset of critical tokens. Motivated by this insight, we introduce Sparse-to-Dense (STD), a novel decoding strategy that integrates two distinct modules: a sparse module that rapidly generates speculative tokens using efficient top-K attention, and a dense module that verifies these tokens in parallel via full self-attention. This collaborative approach accelerates Video-LLMs losslessly, effectively offering a free lunch for video understanding. STD is a plug-and-play solution requiring no fine-tuning or architectural changes and achieves up to a 1.94$\times$ wall time speedup while preserving model performance. It enables a seamless conversion of standard Video-LLMs into sparse counterparts, unlocking efficient long-video processing without sacrificing accuracy.

1 Introduction

Recent advances in Video Large Language Models (Video-LLMs), which combine large language models with video understanding, have achieved exceptional performance on tasks like video question answering and captioning (Lin et al., 2024a; Cao et al., 2024; Zhang et al., 2025a). A common practice in Video-LLMs is representing a video as a sequence of image frames, which results in extremely long token sequences that can strain computational resources. For instance, a 1-hour video sampled at 5-second intervals produces 720 frames, which translates to 141,120 visual tokens in VILA (Lin et al., 2024a). These extremely long token sequences cause Video-LLMs to suffer from high inference latency when processing lengthy videos, making real-time applications challenging.

This latency is primarily introduced by the auto-regressive nature of current Video-LLMs, where each new token must attend to all preceding tokens, creating substantial memory and computational challenges.While mechanisms like key-value (KV) caching are employed to store pre-computed key and value tensors and reduce redundant recomputation, frequent access to the cache imposes heavy demands on memory bandwidth due to the growing amount of KV cache with the increasing sequence length. This significantly reduces the throughput of Video-LLMs. A common approach to addressing this problem is KV cache compression (Du et al., 2024b; Chen et al., 2024b; Lin et al., 2024b; Zhang et al., 2025b) or quantization (Su et al., 2025; Hooper et al., 2024; Liu et al., 2024) at test time. However, these methods introduce discrepancies between training and inference, degrading the performance of LLMs.

In this paper, we aim to build a lossless acceleration method designed specifically for Video-LLMs that preserves the exact output distribution of the original model. Although speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Hou et al., 2025) meets this requirement, it usually requires an extra draft model, which is expensive for Video-LLMs. In contrast, we observe that Video-LLMs exhibit a unique structural property, attention sparsity, which can serve as a training-free and plug-and-play draft model. Specifically, retaining only the top-K KV caches in the attention layers preserves the original predictions for approximately 95% of tokens (empirically verified), suggesting that most attention heads contribute minimally to the final output. Motivated by this observation, we introduce a novel decoding method called Sparse-to-Dense (STD), which leverages the sparse structure of Video-LLMs as its

*Corresponding author: ducx@sea.com

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 734–742

draft model. This design eliminates the need for an extra trained draft model, making STD a plug-and-play solution. We refer to the original Video-LLM as the dense model because it decodes using the full KV cache, whereas the model with top-K attention is termed the sparse model. Both models share identical architectures, differing only in how they compute attention. Therefore, we do not need additional GPU memory to store the sparse model, nor does it require any extra training. The top-K attention in the sparse model boosts decoding speed while sacrificing some token quality, whereas the dense model is slower but guarantees accuracy. We use the sparse model to auto-regressively draft the next γ tokens, while the dense model verifies them in parallel. This approach avoids redundant full KV cache memory and ensures the outputs exactly match those of the original Video-LLM.

We conduct experiments on representative Video-LLMs including LLaVA-OneVision (Li et al., 2024a) and Qwen2-VL (Wang et al., 2024), evaluating them on video understanding benchmarks like MLVU (Zhou et al., 2024) and VideoMME (Fu et al., 2024). Experiment results show that our STD, serving as a tuning-free, plug-and-play solution, achieves up to a $1.94\times$ acceleration of video input processing without any performance degradation. It is immediately deployable, requiring only 20 lines of code to transform an original Video-LLM into a sparse Video-LLM, and it does not require any extra training to deploy the draft model.

2 Observation

In this section, we investigate the disparity in decoded tokens between two configurations of Video-LLMs: 1) sparse top-K KV cache: utilizing only the top-K KV caches based on the highest attention weights; and 2) dense full KV cache: employing the complete set of KV caches. We conduct experiments using the Qwen2-VL-7B (Wang et al., 2024) model on randomly selected samples from MLVU (Zhou et al., 2024), and Video-MME (Fu et al., 2024) datasets. We evaluate the next-token prediction accuracy of the model when employing sparse attention with top-K KV caches. Our findings indicate that the model with sparse attention maintains an average token prediction accuracy exceeding 95%. This high accuracy suggests that for the majority of decoded tokens, only the top-K KV caches are necessary. However, it is important to note that the 95% accuracy is measured per individual token and does not accumulate across multiple tokens. For instance, the accuracy of correctly predicting five consecutive tokens drops to approximately $(95\%)^5 \approx 77\%$.

3 Method

In this section, we present Sparse-to-Dense (STD), a method designed to achieve lossless acceleration for Video-LLMs. We refer to the original model $\mathcal{M}$ as the dense model, as it requires the full KV cache during decoding, while the sparse model $\mathcal{M}_s$ uses sparse attention. Although $\mathcal{M}_s$ is faster, it is somewhat less accurate. Unlike traditional speculative decoding, which relies on an additional draft model, our approach leverages $\mathcal{M}_s$ with the same parameters as $\mathcal{M}$. The only difference is that $\mathcal{M}_s$ loads a reduced KV cache to perform sparse attention, eliminating the need for extra GPU memory to store another model's parameters. In the following subsections, we will detail the decoding procedure and the design of the sparse model.

3.1 Decoding Procedures

In our STD, the sparse model $\mathcal{M}_s$ functions as a draft model to propose potential next γ tokens, while the dense model $\mathcal{M}$ verifies them to derive the final output sequence. Given an input sequence $\{x_0, \cdots, x_{m-1}\}$, consisting of visual and textual tokens , the sparse model $\mathcal{M}_s$ auto-regressively generates γ subsequent token candidates $\{x_m, \cdots, x_{m+\gamma-1}\}$. Because the tokens proposed by the sparse model $\mathcal{M}_s$ might not align with those predicted by the dense model $\mathcal{M}$, it requires the verification of $\mathcal{M}$. The dense model $\mathcal{M}$ verifies all γ proposed tokens in parallel, requiring only a single I/O operation for the full KV cache. Thus, this verification procedure accelerates the process compared with the auto-regressive decoding of $\mathcal{M}$ itself, where each token requires a separate I/O operation. During the verification, $\mathcal{M}$ identifies the first n tokens that align with its predictions, where $0 \leq n \leq \gamma$, and additionally provides a bonus token $\hat{x}_{n+m}$ for free. The verified sequence $\{x_m, \cdots, x_{m+n-1}, \hat{x}_{n+m}\}$ is then appended to the input sequence $\{x_0, \cdots, x_{m-1}\}$ to form the context for the next round of proposal and verification.

3.2 Model with Sparse Attention

Next, we introduce the design of our sparse model $\mathcal{M}_s$. Empirical observations in Section 2 indicate that during most decoding steps, attention scores are predominantly concentrated on a small subset of KV caches, a pattern we term *sparse* attention (also known as top-K attention (Lou et al., 2024)). Only a small fraction of tokens require more evenly distributed *dense* attention. This insight motivates a strategy to selectively apply sparse attention for the majority of tokens and resort to dense attention only when necessary, reducing the I/O overhead of accessing the full KV cache, and thereby improving decoding speed.

Since the number of visual tokens is typically much larger than the number of textual tokens ($m_v \gg m_t$), with m_v often exceeding 10,000 while m_t are usually around 100, our primary focus is on reducing the size of the visual KV cache. To achieve this, we leverage the attention patterns of the textual tokens X_t to identify and select the most relevant KV caches from the visual tokens. Specifically, we analyze the allocation of attention scores when processing the textual tokens $X_t = \{x_{m_v}, \cdots, x_{m-1}\}$ (i.e., the last m_t tokens in the input sequence) to identify which KV pairs of the visual tokens X_v contribute more during the prefilling stage. For each layer l, we calculate the average attention scores directed toward the visual tokens X_v for textual tokens X_t. We then retain only the top-K KV pairs of visual tokens with the highest attention scores. To balance performance and efficiency, we determine the retained K KV caches only during the prefilling stage and avoid the computation-demand dynamic selections in the decoding stage. The selected visual tokens can vary across different layers and attention heads, reflecting the distinct focus of each layer and head in processing the input. The selection of the KV cache of layer l can be formalized as

$$\text{Cache}_s[l] = \text{argTopK}_{x\in X_v}\left(\frac{1}{m_t}\sum_{\hat{x}\in X_t} A_l(\hat{x}, x)\right),$$

where argTopK$(\cdot)$ is an operation that selects the top-K elements indices with the highest values from a given set, k is a predefined hyper-parameter, and $A_l(\hat{x}, x)$ represents the attention score from token $\hat{x}$ to token x in layer l. For models utilizing Grouped Query Attention (GQA) (Ainslie et al., 2023), where the number of query heads equals the number of groups multiplied by the number of KV heads, we directly sum the attention scores within each group to select the top-K KV caches for this head. The KV cache selection operates at the granularity of individual KV heads, allowing each layer or head to retain a distinct subset of caches based on its specific requirements.

3.3 I/O complexity analysis.

In the decoding phase, the I/O complexity of our Sparse-to-Dense decoding method can be analyzed as follows. For the sparse model $\mathcal{M}_s$, which speculatively proposes γ subsequent tokens, the I/O cost involves accessing the selected K visual KV caches and all m_t textual KV caches. Thus, the total I/O for the sparse model is given by: $\text{I/O}_{\text{sparse}} = \gamma \times (K + m_t)$. For the dense model $\mathcal{M}$, which verifies the proposed tokens in parallel, the I/O cost includes accessing the full KV caches of all visual and textual tokens, resulting in: $\text{I/O}_{\text{dense}} = m_v + m_t$. The total I/O for Sparse-to-Dense decoding is therefore: $\text{I/O}_{\text{total}} = \gamma \times (K + m_t) + (m_v + m_t)$, and the average I/O per token is

$$\text{I/O}_{\text{average}} = \frac{\text{I/O}_{\text{total}}}{\alpha \times \gamma} = \frac{\gamma \times (K + m_t) + m_v + m_t}{\alpha \times \gamma},$$

where α ratio of the number of accepted tokens among all proposed tokens. In contrast, the average I/O complexity of vanilla decoding, where each token is generated using full attention, is given by: $\text{I/O}^{\text{vanilla}}_{\text{average}} = m_v + m_t$. When α is sufficiently large, i.e., $\alpha > (K + m_t)/(m_v + m_t) + \gamma^{-1}$, the average I/O per token in our method becomes considerably lower, resulting in improved decoding efficiency. Intuitively, we hope that the ratio between the numbers of the accepted tokens and all proposed tokens is larger than the ratio between the numbers of retained KV pairs and the full KV cache. This can be achieved due to the concentration behavior of attention scores in Section 2. The empirical superiority of our method in the next section verifies this inequality in the realistic setting.

4 Experiment

Baselines. To evaluate the effectiveness of our proposed Sparse-to-Dense decoding, we compare it against the following baselines: 1) Layerskip (Elhoushi et al., 2024): This method utilizes a model with an layer-level early exit mechanism to propose draft tokens. This baseline is inspired by the work of Elhoushi et al. on text-only LLMs, and originally requires additional training. For a fair comparison with our method, we adapt it to Video-LLMs in a tuning-free manner. 2) Streaming (Chen et al., 2024a): This method employs a model with streaming attention (Xiao et al., 2023) to propose

Methods	MLVU		VideoMME-s		VideoMME-m		VideoMME-l	
	Acc. (%)	Speedup	Acc. (%)	Speedup	Acc. (%)	Speedup	Acc. (%)	Speedup
LLaVA-OneVision-7B								
LayerSkip	10.0	0.47×	5.6	0.33×	8.1	0.46×	4.8	0.44×
Streaming	34.7	1.34×	36.4	1.38×	41.0	1.51×	36.2	1.45×
STD (ours)	**47.8**	**1.72**×	**51.8**	**1.82**×	**52.1**	**1.83**×	**52.9**	**1.59**×
Qwen2-VL-7B-Instruct								
LayerSkip	5.2	0.63×	3.7	0.59×	4.9	0.55×	5.7	0.55×
Streaming	53.9	1.61×	52.9	1.32×	59.2	1.36×	59.6	1.36×
STD (ours)	**66.1**	**1.94**×	**71.8**	**1.71**×	**73.4**	**1.62**×	**81.8**	**1.70**×

Table 1: Comparisons of the acceptance rate (Acc.) and wall time speedup of STD and previous draft models. **Bold** denotes the best method. Since all the methods are lossless, we do not report the evaluation of the generated contents.

draft tokens. Similar to LayerSkip, this baseline is derived from the work of Chen et al. on text-only LLMs. To ensure comparability with our approach, we extend its implementation to Video-LLMs.

Datasets and evaluation metrics. We evaluate Sparse-to-Dense on two widely adopted benchmarks: MLVU (Zhou et al., 2024) and VideoMME (Fu et al., 2024). MLVU is specifically designed for long-duration videos, while VideoMME encompasses short, medium, and long-duration videos, providing a comprehensive assessment across various video lengths. For our evaluation, we adhere to the protocols established in previous works on speculative decoding. We report two primary metrics: ***acceptance rate*** of the draft tokens and wall time ***speedup***.

Implementation Details. Our experiments are conducted using widely adopted state-of-the-art Video-LLMs, specifically LLaVA-OneVision (7B) (Li et al., 2024a) and Qwen2-VL (7B) (Wang et al., 2024). We prompt the Video-LLMs to generate chain-of-thought (Wei et al., 2022) responses to enhance their performance. We set the sum of the textual token count m_t and the selected visual KV cache count K to 1024, with a batch size of 8. The number of tokens verified by the dense model $\mathcal{M}_d$ is fixed at $\gamma = 9$. The ablation of hyperparameters can be found in Appendix Section C. Our framework is implemented based on Hugging Face's Transformers library. All experiments are conducted on NVIDIA A100 GPUs with 80 GB of memory, and are repeated three times with different random seeds, and the average results are reported.

Main Results Table 1 summarizes the performance across various reasoning tasks. We have the following findings: 1) *The draft model based on LayerSkip performs worse than that utilizing sparse attention* (e.g., Streaming and STD). The primary reason for this discrepancy is that LayerSkip causes a substantial distributional shift between the draft model and the target model, leading to a low acceptance rate. Although the draft model with layer skipping runs considerably faster than the sparse attention counterparts, this advantage is insufficient to compensate for the overall wall-time speedup loss introduced by layer skipping. 2) *Draft models based on sparse attention generally provide more wall time speedup.* Whether in STD or Streaming, we observe a consistently high acceptance rate. This indicates that, for most of the time, the target model does not require the full KV cache but only a sparsely selected subset cache. However, it is important to note that since LLMs perform autoregressive decoding, an incorrect token can propagate errors to subsequent tokens. Thus verification with the full KV cache is essential. 3) *Our model outperforms the streaming-based draft model*, achieving 62.2% in acceptance length and 1.74× in wall-time speedup on average. This advantage stems from our method's ability to leverage the unique characteristics of Video-LLMs to select important KV cache. As observed in section 2, text-guided video cache selection effectively identifies and retains the most critical cache elements.

5 Conclusion

We introduce STD, a training-free, plug-and-play decoding method that employs sparse top-K attention as the draft model in speculative decoding while leveraging full attention for verification in parallel, ensuring lossless acceleration. Extensive experiments demonstrate that STD significantly outperforms strong baselines that use LayerSkip and Streaming as the draft models. Overall, STD achieves up to a 1.94× walltime speedup while maintaining identical output quality. In the future, we hope to extend our work to accelerate long

CoT Video-LLMs such as QvQ (QwenLM Team, 2024).

Limitation

A notable limitation of our current approach is that all KV caches are still stored in GPU memory (i.e., HBM). While HBM provides the high bandwidth necessary for fast computations, its capacity is inherently limited, which poses a significant bottleneck during inference—especially as model sizes and sequence lengths increase. The limited HBM capacity may lead to restricted batch size.

In the future, a promising solution to this challenge is to offload portions of the KV caches to CPU memory. Although CPU memory typically has lower bandwidth compared to HBM, it offers substantially larger capacity. By developing efficient data transfer and caching strategies, it may be possible to mitigate the HBM bottleneck without sacrificing inference accuracy, thereby enabling more scalable and efficient processing for large Video-LLMs.

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*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Jianjian Cao, Peng Ye, Shengze Li, Chong Yu, Yansong Tang, Jiwen Lu, and Tao Chen. 2024. Madtp: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15710–15719.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Jian Chen, Vashisth Tiwari, Ranajoy Sadhukhan, Zhuoming Chen, Jinyuan Shi, Ian En-Hsu Yen, and Beidi Chen. 2024a. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024b. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. *arXiv preprint arXiv:2403.06764*.

Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, et al. 2024a. Glide with a cape: A low-hassle method to accelerate speculative decoding. *arXiv preprint arXiv:2402.02082*.

Cunxiao Du, Hao Zhou, Zhaopeng Tu, and Jing Jiang. 2024b. Revisiting the markov property for machine translation. *arXiv preprint arXiv:2402.02084*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. 2024. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*.

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

Mukul Gagrani, Raghavv Goel, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. 2024. On speculative decoding for multimodal large language models. *arXiv preprint arXiv:2404.08856*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent YF Tan, and Zhuoran Yang. 2025. Banditspec: Adaptive speculative decoding via bandit algorithms. *arXiv preprint arXiv:2505.15141*.

Zhengmian Hu and Heng Huang. Accelerated speculative sampling based on tree monte carlo. In *Forty-first International Conference on Machine Learning*.

Doohyuk Jang, Sihwan Park, June Yong Yang, Yeonsung Jung, Jihun Yun, Souvik Kundu, Sung-Yub Kim, and Eunho Yang. 2024. Lantern: Accelerating visual autoregressive models with relaxed speculative decoding. *arXiv preprint arXiv:2410.03355*.

Xiaohan Lan, Yitian Yuan, Zequn Jie, and Lin Ma. 2024. Vidcompress: Memory-enhanced temporal compression for video understanding in large language models. *arXiv preprint arXiv:2410.11417*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. 2024a. Vila: On pre-training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26689–26699.

Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. 2024b. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. *arXiv preprint arXiv:2405.05803*.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. 2023. Online speculative decoding. *arXiv preprint arXiv:2310.07177*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.

Chao Lou, Zixia Jia, Zilong Zheng, and Kewei Tu. 2024. Sparser is faster and less is more: Efficient sparse attention for long-range transformers. *arXiv preprint arXiv:2406.16747*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949.

QwenLM Team. 2024. Qvq-72b preview. https://qwenlm.github.io/blog/qvq-72b-preview/.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. 2024. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*.

Dingjie Song, Wenjun Wang, Shunian Chen, Xidong Wang, Michael Guan, and Benyou Wang. 2024. Less is more: A simple yet effective token reduction method for efficient multi-modal llms. *arXiv preprint arXiv:2409.10994*.

Zunhai Su, Wang Shen, Linge Li, Zhe Chen, Hanyu Wei, Huangqi Yu, and Kehong Yuan. 2025. Akvq-vl: Attention-aware kv cache adaptive 2-bit quantization for vision-language models. *arXiv preprint arXiv:2501.15021*.

Ziteng Sun, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. 2024a. Optimal block-level draft verification for accelerating speculative decoding. *arXiv preprint arXiv:2403.10444*.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2024b. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. 2024. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices. *arXiv preprint arXiv:2406.02532*.

Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. 2024. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. *arXiv preprint arXiv:2410.01699*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuxin Wen, Qingqing Cao, Qichen Fu, Sachin Mehta, and Mahyar Najibi. 2024. Efficient vision-language models by summarizing visual tokens into compact registers. *arXiv preprint arXiv:2410.14072*.

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

Linli Yao, Lei Li, Shuhuai Ren, Lean Wang, Yuanxin Liu, Xu Sun, and Lu Hou. 2024. Deco: Decoupling token compression from semantic abstraction in multimodal large language models. *arXiv preprint arXiv:2405.20985*.

Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, et al. 2025a. Videollama 3: Frontier multimodal foundation models for image and video understanding. *arXiv preprint arXiv:2501.13106*.

Xuan Zhang, Fengzhuo Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. 2025b. Light-transfer: Your long-context llm is secretly a hybrid model with effortless adaptation. In *Workshop on Reasoning and Planning for Large Language Models*.

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

A Preliminary

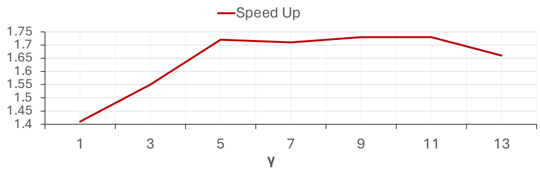
Speculative Decoding We first formalize our notation and provide a brief overview of the speculative decoding in autoregressive LLMs, which is the key background knowledge for our method. We represent the input sequence for a Video-LLM as a combination of visual tokens and textual tokens. Specifically, the visual tokens are denoted as $X_v = \{x_0, \cdots, x_{m_v-1}\}$, and the textual prompt is denoted as $X_t = \{x_{m_v}, \cdots, x_{m-1}\}$. Here, m_v is the number of visual tokens, m_t is the number of textual tokens, and the total input sequence length is $m = m_v + m_t$. The key and value cache for token x_i are represented by K_{x_i} and V_{x_i}, respectively.

Inference of Auto-regressive Models. The inference stage of auto-regressive models, e.g., Video-LLMs, can be divided into two stages: 1) *prefilling*: The video LLM processes the input sequence, which includes both visual tokens X_v and textual tokens X_t, in an autoregressive and parallel manner. For each token x_i in the combined input $\{X_v, X_t\}$, the model computes and stores the corresponding KV cache entries. This stage effectively encodes the input sequence and prepares the model for generating a response. The output of this stage is the first token x_m of the model's response. 2) *decoding*: After prefilling, the model enters the decoding phase, generating output tokens sequentially. At each decoding step $j = m + 1, m + 2, \cdots$, the video LLM generates a new token x_j based on the KV cache from all prior tokens. After generating, the KV cache is updated with each newly generated token. This process continues iteratively until a stopping criterion is met, such as reaching an end-of-sequence token or hitting a maximum token limit.

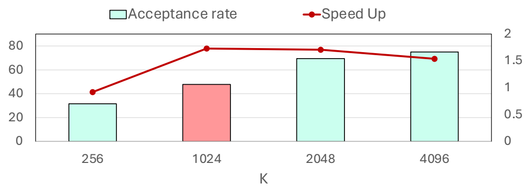
B Related Works

Sparse Attention in MLLMs Normally, an image or a video frame is represented as a large number of tokens in MLLMs, e.g., 196 visual tokens per image in VILA (Lin et al., 2024a), which significantly impacts the computational and storage during model training and inference. Visual token compression aims to reduce the number of visual tokens to address it directly. The majority of visual token compression methods either train from scratch or perform additional training based on existing models. For example, some image-based MLLMs rely on vision-language alignment (Cao et al., 2024; Yao et al., 2024; Song et al., 2024) or aggressively removing all visual tokens after a certain layer (Wen et al., 2024), while methods designed for video-based MLLMs consider the unique characteristics of video, such as employing memory mechanisms (Lan et al., 2024) or compressing tokens along spatial and temporal dimensions sequentially (Shen et al., 2024). A smaller portion of works study the test-time (training free) visual token compression for accelerating the inference procedure. FastV (Chen et al., 2024b) performs pruning by analyzing the attention pattern from shallow layers and deep layers, while another approach directly applies full visual token removal during the inference stage (Lin et al., 2024b). In our method, STD, the design of the drafter model is related to training-free visual token compression techniques. However, these previous methods inevitably impact the original model's performance. In contrast, we propose to utilize visual token compression as a drafter model to achieve lossless inference acceleration.

Speculative Decoding Speculative decoding is proposed by (Leviathan et al., 2023) and (Chen et al., 2023) to accelerate the inference of LLMs, where the throughput of LLMs is improved $2 \sim 3$ times without sacrificing the performance. The algorithm consists of two stages: drafting and verification. The drafting stage adopts a small model (drafter) to generate a long sequence of possible future tokens swiftly, while the verification stage accepts a part of the tokens predicted in the drafting stage in a token-by-tone manner. The follow-up improves the speculative decoding from these two perspectives. Specinfer (Miao et al., 2024), Eagle (Li et al., 2024b) and Medusa (Cai et al., 2024) propose to train a drafter to generate tokens with a tree structure, and the verification is conducted on the tree in a branch-by-branch manner. Hu and Huang (Hu and Huang) also organize the draft tokens as a tree, but they verify the tokens in a branch as a whole. Glide (Du et al., 2024a) generates draft tokens as an unbalanced tree, which alleviates the burden of the drafter while achieving significant acceleration. SpecTr (Sun et al., 2024b) views speculative decoding from the optimal transport view and proposes to verify a batch of draft tokens jointly. They show that the proposed algorithm is optimal up to a multiplicative factor. Sun et al. (Sun et al., 2024a) boot the acceleration by a joint verification



(a)



(b)

Figure 1: Effect of K and γ on MLVU using LLaVA-OneVision-7B.

of a single draft trajectory. Instead of using a token-by-token manner, they accept the draft sentences as a whole. Lie et al. (Liu et al., 2023) proposes to update the parameters of drafters in an online manner, which is shown to be effective in various applications. MagicDec (Chen et al., 2024a) analyzes the speculative decoding in the long-context setting with an emphasis on the FLOPS and memory. SpecExec (Svirschevski et al., 2024) focuses on a special setting where the LLMs are offloading their parameters. Several works (Gagrani et al., 2024; Jang et al., 2024; Teng et al., 2024) study the speculative decoding of MLLMs. However, they focus either on the image understanding problem or the image generation problem. In contrast, our work is the first to study video generation acceleration via speculative decoding.

C Ablation Stuidies

We also conducted additional experiments to analyze the impact of hyperparameters (γ and K) on model performance. As shown in Figure 1a, we can see that as $gamma$ increases, the speed up gradually improves. This improvement is because the sparse model makes accurate predictions, which allows the computational overhead to be spread out over more tokens. However, when $gamma$ reaches 13, the speed up starts to decline because the model's accuracy in correctly predicting 13 consecutive tokens is insufficient. At the same time, as shown in Figure 1b, when K is small, the acceptance rate is low, resulting in a lower speed up. In contrast, when K is large, the sparse model is not as fast, which also leads to a reduced speed-up.