Fewer Tokens, Fewer Layers: Efficient Vision Token Pruning and On-Policy Distillation to Accelerate VLMs

Shuai Wang[1], **Shitong Shao**[1], **Qi Xuan**[2], **Zhaowei Zhu**[3], **Jiaheng Wei**[1]
[1] The Hong Kong University of Science and Technology (Guangzhou)
[2] Zhejiang University of Technology [3] Docta.ai
swang190@connect.hkust-gz.edu.cn

Abstract

Vision-language models (VLMs) have achieved remarkable progress. However, processing high-resolution images and long videos generates a massive number of vision tokens, creating a severe inference bottleneck due to the quadratic complexity of attention mechanisms. Previous efforts to mitigate this issue have primarily focused on accelerating the prefill stage by reducing the number of vision tokens, often yielding limited practical speedups while neglecting the increasingly time-consuming decoding stage when the generated response is long. In addition, existing approaches heavily rely on attention maps or similarity maps that are not compatible with efficient attention implementations. To comprehensively address these limitations, we first propose **FlashPruner**, a lightweight module that selects the most helpful vision tokens without relying on attention maps, significantly reducing prefill time. For the decoding stage, we adopt self-speculative decoding and theoretically prove that on-policy distillation guarantees a lower bound for the speculative acceptance rate. Guided by this formulation, we introduce **LoFT**, an on-policy distillation method that effectively trains a highly reliable and efficient draft model including FlashPruner and early layers of the language model. Extensive experiments demonstrate that after training with LoFT, FlashPruner achieves substantial inference speedups while preserving task performance. Notably, on MVBench, FlashPruner improves the time-to-first-token by 1.5× and decoding speed by 2×, maintaining 96.8% relative accuracy with only 10% vision tokens.

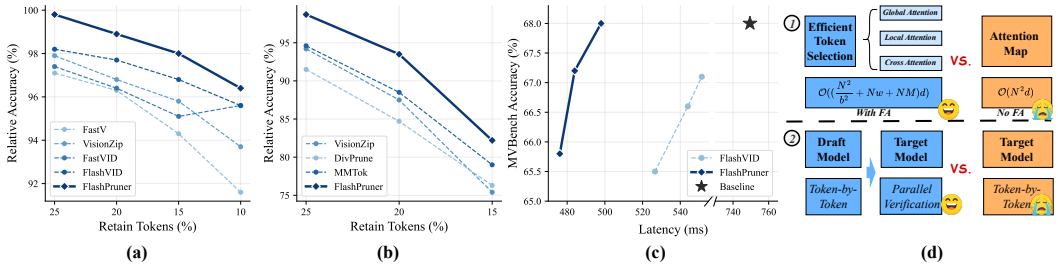


Figure 1: Overall results on video **(a)** and image **(b)** benchmarks of different approaches with various retained token ratios. **(c)** presents latency and performance comparison on the proposed FlashPruner and FlashVID under different retained tokens. **(d)** shows the highlight of **FlashPruner** in prefill (①) and decoding stage (②) during inference. FA denotes FlashAttention.

1 Introduction

Vision-language models (VLMs) have made significant progress on various vision-language tasks, such as visual reasoning, document understanding, and video understanding [4, 29]. While powerful

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and effective, VLMs typically have large parameter counts. As a result, deploying VLMs in real-world applications is challenging due to the substantial computational and GPU memory resources required. Inference acceleration for VLMs has become an important practical research direction.

The typical VLMs include three parts: a vision encoder such as CLIP [37], a projector and a language model (LM) [6, 53]. Given vision inputs, the vision encoder and projector first encode them into a sequence of vision tokens. After that, combining the embeddings of text tokens, they are fed into the LM to generate text tokens. The computational complexity of attention [42], the core module of transformer architectures, increases quadratically with the number of tokens and requires substantial computation and GPU memory, especially for long sequences. Therefore, reducing the number of tokens, also known as token pruning, is an effective way to accelerate VLMs by shortening the sequence length for attention computation [9, 55, 15]. The most widely studied approach for VLMs is vision token pruning, because the number of vision tokens is usually greater than textual tokens and often contains more redundant information, especially in high-resolution images and long videos.

Motivated by this, a series of approaches have been proposed to reduce vision tokens by dropping redundant tokens using attention map information [9, 55, 2], spatiotemporal relationships and redundancy in videos [15] to accelerate the inference of VLMs. Despite these effective approaches, they suffer from two limitations. The **first** is that many approaches rely on the attention map to prune vision tokens. Although effective, fetching the attention map is not compatible with FlashAttention [11], which is widely used in modern large-scale model training and inference. This leads to a failure to achieve speedups comparable to their theoretical gains. In addition, attention maps also suffer from internal bias toward the end of the visual sequence [8]. **Second**, almost all methods focus on acceleration at the *prefill* stage. However, as we show in experiments, the time cost of the *decoding* stage is not negligible, especially in chain-of-thought prompts [48] (CoT) and reasoning models [3].

To mitigate the prefill bottleneck, we introduce **FlashPruner**, a lightweight pruning block placed after the VLM projector to select task-relevant vision tokens. Unlike attention- or similarity-based pruning methods, FlashPruner does not require accessing attention maps and is therefore compatible with efficient attention kernels on modern GPUs. To further accelerate decoding, we first analyze self-speculative decoding [14, 22] and show that, under the standard stochastic acceptance rule, controlling the divergence between the early-exit draft distribution and the full-model target distribution yields a lower bound on the expected acceptance rate (see Theorem 1). This motivates training a stronger draft model rather than directly applying early-exit decoding. We therefore propose fewer **l**ayers and t**o**kens **f**ine-**t**uning (**LoFT**), an on-policy distillation method that aligns both the early-exit draft branch and the full-depth pruned-token branch with the teacher on student-generated prefixes. LoFT reduces the gap between teacher-forcing training and autoregressive inference, and improves the model's ability to perform tasks from pruned vision tokens. At inference time, the trained early layers act as the draft model and the full language model serves as the target model, enabling effective self-speculative decoding in the decoding stage.

Extensive experiments on multiple image and video understanding benchmarks validate the effectiveness of the proposed method. Notably, on image understanding benchmarks, FlashPruner preserves 82.2% relative accuracy on five benchmarks with retained 5% vision tokens for Qwen2.5-VL-7B [4], significantly outperforming previous state-of-the-art approaches. As for video understanding tasks, with only 10% vision tokens, FlashPruner maintains 96.4% relative accuracy average over four benchmarks. In terms of efficiency, at the prefill stage, FlashPruner accelerates the prefill stage by 5.3$\times$ for the language model with negligible token pruning time cost. At decoding, FlashPruner achieves 2$\times$ decoding speedup while most previous methods do not accelerate it. These results demonstrate significant improvements over previous state-of-the-art methods in terms of both accuracy and speed.

To summarize, our contributions are as follows:

• We identify that existing vision token pruning methods often rely on full attention or similarity maps, limiting practical efficiency. We propose FlashPruner, a hybrid sparse-attention pruner that efficiently selects informative vision tokens and accelerates both prefill and decoding.

• We show that on-policy draft–target alignment yields a lower bound on the expected acceptance rate under bounded KL divergence and transfer assumptions. Based on this insight, we propose LoFT to train effective early-layer draft models for self-speculative decoding.

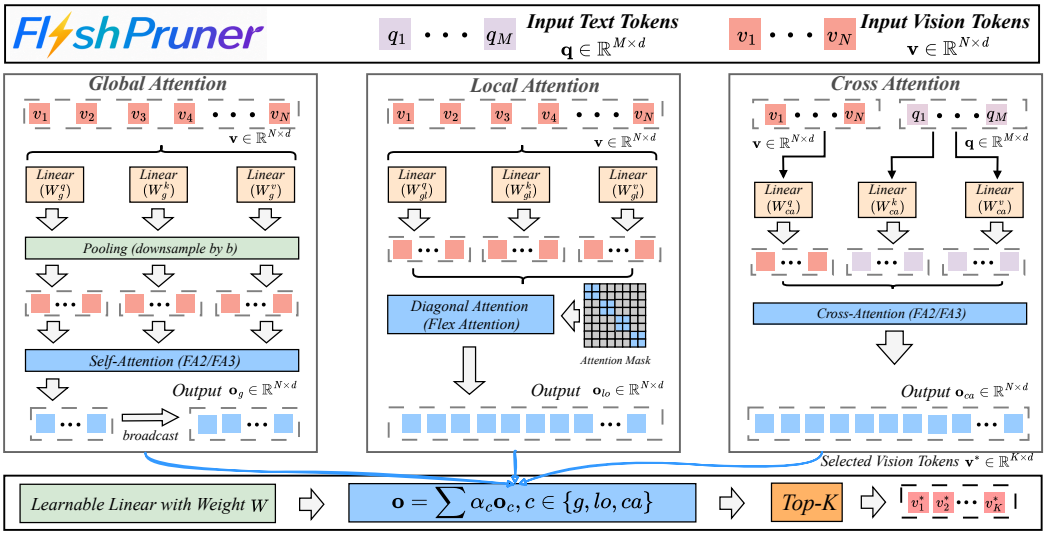


Figure 2: **The architecture of FlashPruner**. FlashPruner includes three parts: global attention modeling, local attention modeling and cross attention between text query and vision tokens. By FlashPruner, we obtain the score for each vision token in $\mathbf{v}$ and select the top-K tokens $\mathbf{v}^*$ to feed the language model. α_c is a learnable parameter. See Sec. 3.1 for details.

• Experiments show that FlashPruner with LoFT achieves 5.3× prefill and 2× decoding speedups, while retaining 82.2% relative accuracy on image benchmarks with 5% tokens and 96.4% on video benchmarks with 10% tokens on Qwen2.5-VL-7B.

2 Preliminaries

VLM architecture and inference. Modern VLMs [29, 4, 47] are usually comprised of a vision encoder $f(\cdot)$, a projector $g(\cdot)$, and a language model p_θ parameterized by θ. During inference, VLMs first encode visual inputs $\mathbf{x}$ as a sequence of N vision tokens $\mathbf{v} = [v_1, \ldots, v_N] = g(f(\mathbf{x})) \in \mathbb{R}^{N \times d}$, where d is the feature dimension. The embedding $\mathbf{q}$ of the text input is usually a sequence of M text tokens, i.e., $\mathbf{q} = [q_1, \ldots, q_M] \in \mathbb{R}^{M \times d}$. These two groups of tokens are concatenated and fed into p_θ to generate a sequence of text tokens $\mathbf{y} = [y_1, \ldots, y_L] \in \mathbb{R}^{L \times V}$ where V is vocabulary size and L is the length of generated text tokens. In practice, the iterative generation process is composed of two stages: prefill and decoding [64]. At the prefill stage, the language model p_θ deals with the sequence $[\mathbf{v}, \mathbf{q}]$ and generates the first token y_1 in one step. In the decoding stage, each decoding step only generates one new token, i.e., at the ℓ-th step, $y_\ell \sim p_\theta(\cdot \mid \mathbf{v}, \mathbf{q}, \mathbf{y}_{<\ell})$.

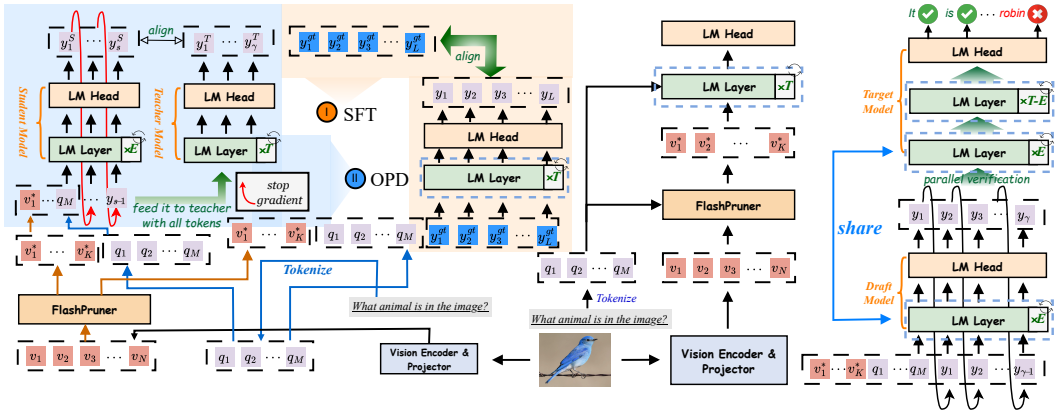
The distinct computational characteristics of the two phases lead to significant differences. When dealing with multiple tokens, the prefill stage tends to be computation-bound. The time cost of prefill highly depends on the number of input tokens. In contrast, LMs usually generate one token per step in the decoding stage, making the process constrained by the memory bandwidth of GPUs.

Speculative decoding. To improve the decoding speed of autoregressive models, speculative decoding [22] is proposed to improve decoding efficiency without performance degradation. The standard speculative decoding includes two models: a draft model and a target model. At each decoding step, the draft model first generates γ draft tokens from a given input in an autoregressive manner. After that, the target model validates the generated γ draft tokens in a *parallel* way to determine whether the generated tokens by the draft should be accepted.

3 Method

3.1 Efficient and Effective Token Pruning using FlashPruner

The prefill stage is computation-bound, therefore reducing the number of input tokens significantly accelerates prefill. Previous work mainly focuses on post-hoc approaches to reduce the number



(a) Training *(b) Prefill* *(c) Decoding*

Figure 3: **Overview of training and inference stage.** (a) demonstrates the pipeline of two-stage **training**. In the first stage, we fine-tune FlashPruner with an additional labeled dataset to initialize FlashPruner. In the second stage, we distill from outputs of all vision tokens to make the language model understand selected vision tokens by FlashPruner. We randomly drop the later $T - E$ layers as the student model and feed it selected vision tokens. We use the frozen language model with T layers as the teacher and feed it all vision tokens. During **inference**, in the prefill stage, FlashPruner selects the most important K vision tokens to reduce the number of vision tokens as shown in (b). During the decoding stage (c), we use the first E layers of the language model as the draft model and the full T-layer model as the target model for self-speculative decoding.

of vision tokens by selecting or merging them. The attention map plays an important role in their methods [9, 15]. However, fetching the attention map is not compatible with FlashAttention [11], which has been widely adopted to effectively accelerate attention computation and has become the de facto standard for attention implementation. In addition, the attention map also suffers from internal bias, which leads to unreliable token selection [8].

To address the above issues, we design FlashPruner h parameterized by φ to reduce the number of vision tokens. Our goal is to design efficient and effective token selection modules to reduce the number of vision tokens while being highly efficient on modern hardware. The overview architecture of FlashPruner is presented in Fig. 2. Our FlashPruner includes three parts: global attention modeling to learn global relevance among all vision tokens, local attention to capture adjacent vision token redundancy, and cross attention to analyze relevance between the input text query and vision tokens. We use W with superscript $\{q, k, v\}$ to denote learnable weights for query, key, and value in the attention block, and different subscripts $\{g, lo, ca\}$ denote weights for different modules.

The global attention module is designed to learn block-wise information with minimal overhead to capture the information of all vision tokens. By aggregating sequential blocks into block-level tokens, we obtain compressed vision tokens that capture the information of the entire visual inputs. Given a sequence of vision tokens $\mathbf{v}$ and pooling size b, we first project $\mathbf{v}$ to query, value, and key, and downsample them by b to reduce the number of tokens. After that, the downsampled vision sequence is processed through a standard self-attention mechanism. The computation cost is $\mathcal{O}(\frac{N^2}{b^2}d)$, which is more efficient compared with computing attention with the original vision sequence $\mathbf{v}$. To be specific, the computation is formulated as follows

$$\begin{aligned} \widetilde{\mathbf{Q}}_g = W_g^q \mathbf{v},\ \widetilde{\mathbf{K}}_g = W_g^k \mathbf{v},\ \widetilde{\mathbf{V}}_g = W_g^v \mathbf{v}, \\ \mathbf{Q}_g, \mathbf{K}_g, \mathbf{V}_g = \text{pooling}(\widetilde{\mathbf{Q}}_g), \text{pooling}(\widetilde{\mathbf{K}}_g), \text{pooling}(\widetilde{\mathbf{V}}_g), \\ \widetilde{\mathbf{o}}_g = \text{softmax}\left(\mathbf{Q}_g \mathbf{K}_g^T / \sqrt{d}\right) \mathbf{V}_g,\ \mathbf{o}_{(i-1)b+j} = \widetilde{\mathbf{o}}_i, \quad i = 1, 2, \ldots, N/b, \quad j = 1, 2, \ldots, b. \end{aligned} \tag{1}$$

Besides global patterns, local patterns typically capture the redundancy between adjacent tokens in videos and images. We adopt local attention to capture it. Specifically, we only maintain a block with size w and calculate self-attention in the block as follows

$$\begin{aligned} \mathbf{Q}_{lo} = W_{lo}^q \mathbf{v},\ \mathbf{K}_{lo} = W_{lo}^k \mathbf{v},\ \mathbf{V}_{lo} = W_{lo}^v \mathbf{v}, \\ \mathbf{o}_{lo} = \text{softmax}\left(\mathbf{Q}_{lo} \mathbf{K}_{lo}^T / \sqrt{d} + \mathbf{M}\right) \mathbf{V}_{lo},\ M_{ij} = \begin{cases} 0, & \left\lfloor \frac{i-1}{w} \right\rfloor = \left\lfloor \frac{j-1}{w} \right\rfloor \\ -\infty, & \text{otherwise} \end{cases}. \end{aligned} \tag{2}$$

The local attention introduces additional computation $\mathcal{O}(Nwd)$ in the attention, which is also smaller than computation on the whole sequence.

To model the similarity between the input text query and vision tokens, we introduce cross attention to obtain query-aware importance of vision tokens as follows

$$\mathbf{Q}_{ca} = W_{ca}^{q}\mathbf{v},\ \mathbf{K}_{ca} = W_{ca}^{k}\mathbf{q},\ \mathbf{V}_{ca} = W_{ca}^{v}\mathbf{q},\ \mathbf{o}_{ca} = \text{softmax}\left(\mathbf{Q}_{ca}\mathbf{K}_{ca}^{T}/\sqrt{d}\right)\mathbf{V}_{ca}. \tag{3}$$

It brings little additional computation $\mathcal{O}(NMd)$ due to $M \ll N$ in most cases.

The final output is acquired by $\mathbf{o} = \sum_{c\in\{ca,lo,g\}} \alpha_c \mathbf{o}_c$ where $\alpha_c \in (0,1)$ is a learnable parameter to control the importance of different parts, and we use a linear layer W to get the score for each token $\mathbf{s} = W\mathbf{o} \in \mathbb{R}^N$. Then we choose K tokens that have the highest K scores. First, we select the top-K scores in $\mathbf{s}$ and get the index set $\mathcal{I}_K \subseteq \{1, 2, \ldots, N\}$, which is the set of indices of the top K highest scores. We then select the tokens and get $\mathbf{v}^* = [v_1^*, \ldots, v_K^*] = \mathbf{v}[\mathcal{I}_K, :]$. $\mathbf{v}[\mathcal{I}_K, :]$ denotes selecting the tokens from $\mathbf{v}$ corresponding to the indices in $\mathcal{I}_K$. After vision token pruning, we concatenate the selected vision tokens $\mathbf{v}^* \in \mathbb{R}^{K\times d}$ and the embedding of the text query $\mathbf{q} \in \mathbb{R}^{M\times d}$ to feed forward to the LLM p_θ, as shown in Fig. 3(b). FlashPruner introduces little parameters and we analyze efficiency of FlashPruner in Sec. E.3.

3.2 Self-speculative Decoding for Decoding Acceleration

Most previous methods mainly focus on acceleration in the prefill stage. However, based on our experiments, the time cost of the decoding stage cannot be ignored, especially for long responses. We adopt self-speculative decoding [14] to improve decoding speed without additional parameters, unlike [40, 7] or separate draft models [27].

The overview of the decoding stage during inference is presented in Fig. 3(c). For LMs with T layers, we adopt self-speculative decoding with early exit layers [14]. At the decoding stage, we denote the first E layers with an LM head as the *draft* model and the entire language model as the *target* model. During the decoding phase, we first use the draft model to perform standard autoregressive inference to generate γ draft tokens. Then, we leverage the target model to predict the next token for each draft token in a *single forward pass* for verification. Finally, we sample tokens following the principles in [22] (see Algorithm 1 in [22]). The key difference compared with LayerSkip [14] lies in the training approach: we use on-policy distillation, as shown below, to remove the gap between teacher-forcing training and autoregressive inference, unlike [14].

Note that the verification stage critically only requires computing the remaining $T - E$ layers of the model that were not used in the draft stage. We re-use the KV cache of the first E layers. To avoid redundant computation, we also save the query vectors in the E layers for verification to directly continue from layer E to the last layer T. Saving query vectors only in the E-th layer introduces little cache storage requirements, and a detailed analysis is included in Sec. E.3.

3.3 Training

The training process is designed around the two inference-time roles of FlashPruner. It must first select a compact set of vision tokens that preserves task-relevant information, and the resulting pruned-token VLM must then support self-speculative decoding, where an early-exit draft model proposes draft tokens and the full-layers target model verifies them. We therefore use a two-stage procedure: a supervised warm-up stage that initializes FlashPruner, followed by LoFT, an acceptance-guided on-policy distillation stage derived from the lower bound below. The training process is illustrated in Fig. 3(a).

Stage 1: Warm-up. We first initialize FlashPruner while keeping the original VLM fixed. Given visual input $\mathbf{x}$, text query $\mathbf{q}$, and label sequence $\mathbf{y}$, the vision encoder and projector produce vision tokens $\mathbf{v} = g(f(\mathbf{x}))$. The pruner h_φ selects K vision tokens from $\mathbf{v}$, and the frozen language model predicts the response conditioned on the selected vision tokens and the query. We optimize only φ with the supervised objective

$$\mathcal{L}_{\text{SFT}}(\varphi) = -\sum_{\ell=1}^{|\mathbf{y}|} \log p_\theta\left(y_\ell \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{<\ell}\right). \tag{4}$$

Table 1: **Comparison on Qwen2.5-VL-7B.** Avg. are computed over first 5 datasets following [12]. We **highlight** the best result with the same retained tokens.

Method	**GQA**	**MMB**	**MME**	**POPE**	**VQA$^{\text{Text}}$**	**SQA**	**OCRBench**	**Avg.**
Metric	Acc.	Acc.	P+C	F1	Acc.	Acc.	Acc.	Rel. Acc.(%)
100% Tokens								
Qwen-2.5-VL-7B	60.48	83.25	2327	86.16	77.72	87.46	83.80	100
Retain 20% Tokens								
VisionZip	56.80	80.33	2174	83.38	70.43	84.23	59.50	94.2
DivPrune	56.70	76.98	2163	80.59	65.86	80.91	48.10	91.5
MMTok	58.09	79.30	2217	82.38	70.49	81.61	59.60	94.6
FlashPruner	**60.20**	**82.91**	**2368**	**85.43**	**72.42**	**86.81**	**61.20**	**98.7**
Retain 10% Tokens								
VisionZip	52.47	75.60	2003	78.90	63.78	82.30	36.90	87.5
DivPrune	53.43	72.85	1957	74.99	59.59	79.57	37.30	84.7
MMTok	55.09	74.74	2051	78.75	63.90	80.47	43.60	88.5
FlashPruner	**58.10**	**78.40**	**2184**	**83.14**	**67.26**	**84.88**	**45.30**	**93.5**
Retain 5% Tokens								
VisionZip	46.28	67.53	1677	66.38	54.49	79.57	19.70	75.4
DivPrune	49.01	65.89	1739	68.45	52.02	77.05	24.90	76.3
MMTok	50.66	65.89	1796	71.35	55.95	77.19	30.70	79.0
FlashPruner	**52.10**	**70.18**	**1832**	**74.63**	**58.64**	**82.40**	**33.80**	**82.2**
0 Token								
Qwen-2.5-VL-7B	31.84	20.10	935	0.00	38.93	71.10	1.80	33.8

We randomly sample the retention ratio r from $\{5\%, 10\%, 15\%, 20\%\}$ and set $K = \lfloor rN \rfloor$ to support different r at the inference time. This stage does not yet optimize speculative decoding. Its role is to prevent the later on-policy stage from starting with random or unstable visual token selection.

Acceptance optimization for self-speculative decoding. After token pruning, the main question is how to make the first E layers a useful draft model. Let T be the total number of language-model layers. During self-speculative decoding, the first E layers generate γ draft tokens autoregressively, and the full T-layer model verifies these draft tokens in parallel. For a draft position i, let q_i be the draft distribution produced by the early-exit model and let p_i be the verifier distribution produced by the full model, both conditioned on the same pruned vision tokens and the same accepted prefix. Under the standard stochastic speculative decoding rule, the expected accepted prefix length is controlled by the distance between q_i and p_i. This gives the following training target.

Theorem 1 (Lower bound of acceptance rate)**.** *Let A_γ denote the accepted prefix length among γ draft tokens, and define the acceptance rate as A_γ/γ. Assume that self-speculative decoding uses the standard stochastic acceptance rule after all decoding transformations. For each draft position $i \in \{1, \dots, \gamma\}$, let q_i and p_i denote the draft and target next-token distributions conditioned on the same active decoding context. If there exists $\varepsilon_{\text{pos}} \geq 0$ such that, whenever position i is reached,*

$$\mathrm{KL}(q_i \| p_i) \leq \varepsilon_{\text{pos}}, \tag{5}$$

then the expected acceptance rate satisfies

$$\frac{\mathbb{E}[A_\gamma]}{\gamma} \geq \max\left\{0,\ 1 - \frac{\gamma+1}{2}\sqrt{\frac{\varepsilon_{\text{pos}}}{2}}\right\}. \tag{6}$$

The theorem shows that improving self-speculative decoding amounts to reducing the per-position divergence between the early-exit draft distribution and the full-depth target distribution. The proof is provided in Sec. F in the Appendix. In our setting, this means making

$$q_i = p_E(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{<i}) \quad \text{close to} \quad p_i = p_T(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{<i}). \tag{7}$$

A direct draft-target matching loss is possible, but it is not ideal: both p_E and p_T are conditioned on pruned vision tokens, so the two branches may become mutually consistent while still drifting away

Table 2: **Results on video understanding tasks with Qwen2.5-VL-7B**. Results of other baselines are from FlashVID [15] and we report accuracy for all benchmarks. The best result with the same retained tokens is highlighted in **bold**.

Method	VideoMME				EgoSchema		LongVideo Bench	MVBench	Avg.	
	Short	Medium	Long	Overall	Subset	Total			Acc.	Rel. Acc.(%)
100% Tokens										
Qwen-2.5-VL-7B	72.6	61.4	49.9	61.3	60.2	58.3	58.9	68.0	61.6	100.0
Retain 25% Tokens										
FastV	71.2	57.8	51.6	60.2	60.2	57.2	54.6	67.4	59.8	97.1
VisionZip	70.7	57.9	51.2	59.9	58.6	57.0	57.1	67.3	60.3	97.9
FastVID	71.2	57.8	49.9	59.6	58.2	56.7	58.0	65.5	60.0	97.4
FlashVID	71.1	58.7	49.1	59.6	59.4	57.2	58.1	67.1	60.5	98.2
FlashPruner	**71.8**	**60.7**	**52.0**	**61.5**	**60.6**	**58.2**	**58.3**	**68.0**	**61.5**	**99.8**
Retain 20% Tokens										
FastV	69.4	57.0	51.2	59.2	**60.2**	57.1	54.2	66.8	59.3	96.3
VisionZip	69.6	57.2	50.2	59.0	58.6	56.6	56.3	66.4	59.6	96.8
FastVID	69.9	56.3	49.7	58.6	57.2	56.4	57.8	64.7	59.4	96.4
FlashVID	70.4	58.6	49.7	59.6	59.2	56.8	58.1	66.5	60.2	97.7
FlashPruner	**71.6**	**59.4**	**51.8**	**60.9**	**60.2**	**57.6**	**58.1**	**67.2**	**61.0**	**98.9**
Retain 15% Tokens										
FastV	67.4	55.2	51.1	57.9	59.6	56.5	52.2	65.9	58.1	94.3
VisionZip	68.8	56.7	49.1	58.2	56.2	55.7	56.3	66.0	59.0	95.8
FastVID	68.2	56.9	49.4	58.2	56.6	56.0	56.8	63.6	58.6	95.1
FlashVID	69.8	57.1	49.3	58.7	57.6	56.4	56.8	66.6	59.6	96.8
FlashPruner	**70.4**	**57.8**	**51.3**	**59.8**	**60.0**	**57.0**	**57.6**	**67.2**	**60.4**	**98.0**
Retain 10% Tokens										
FastV	63.7	54.7	49.3	55.9	58.6	54.9	51.1	63.6	57.3	91.6
VisionZip	67.0	54.7	47.6	56.4	55.8	55.5	54.5	64.3	57.7	93.7
FastVID	66.3	53.6	49.0	56.3	56.0	55.6	55.4	62.3	57.4	93.2
FlashVID	68.1	54.7	49.0	57.3	57.4	55.9	57.1	65.5	58.9	95.6
FlashPruner	**68.9**	**55.0**	**49.8**	**57.9**	**59.2**	**56.0**	**58.0**	**65.8**	**59.4**	**96.4**

from the original full-token model. To preserve the original model behavior, we introduce a frozen full-token teacher distribution $r_i = p^*(\cdot \mid \mathbf{v}, \mathbf{q}, y_{<i})$, where p^* is the original language model with all vision tokens. By the triangle inequality of total variation,

$$\mathrm{TV}(q_i, p_i) \le \mathrm{TV}(q_i, r_i) + \mathrm{TV}(p_i, r_i), \tag{8}$$

and Pinsker's inequality further relates each term to a KL divergence to the teacher. Therefore, aligning both the early-exit branch and the full-depth pruned-token branch to the same full-token teacher indirectly reduces the draft-target gap in the theorem while preserving task behavior.

Stage 2: LoFT with on-policy teacher-bridge distillation. Motivated by the bound, LoFT jointly updates the pruner h_φ and the language model p_θ. For each image-query pair, the current student first generates an on-policy response with selected vision tokens $\mathbf{y}_s \sim p_\theta(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q})$. We then distill the student on its own generated prefixes, rather than on ground-truth prefixes, to reduce the mismatch between teacher-forced training and autoregressive inference. Along each prefix $y_{s,<\ell}$, the frozen teacher p^* receives all vision tokens $\mathbf{v}$, while the student receives only selected vision tokens $h_\varphi(\mathbf{v})$. To train both the draft and target branches, LoFT randomly drops the last $T - E$ layers of the student with probability ϵ. The student distribution used in the loss is

$$p'_\theta(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{s,<\ell}) = \begin{cases} p_E(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{s,<\ell}), & \text{with probability } \epsilon, \\ p_T(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{s,<\ell}), & \text{with probability } 1-\epsilon. \end{cases} \tag{9}$$

The LoFT objective is

$$\mathcal{L}_{\mathrm{LoFT}}(\theta, \varphi) = \sum_{\ell=1}^{|\mathbf{y}_s|} \mathrm{KL}\left(p'_\theta(\cdot \mid h_\varphi(\mathbf{v}), \mathbf{q}, y_{s,<\ell}) \| p^*(\cdot \mid \mathbf{v}, \mathbf{q}, y_{s,<\ell})\right). \tag{10}$$

Global	Local	Cross	MVBench Acc.	EgoSchema Acc.	POPE F1
✔			64.8	58.8	71.45
	✔		65.2	57.6	73.84
		✔	65.6	57.4	65.32
✔	✔	✔	65.8	59.2	74.63
All tokens			68.0	60.2	86.16

Table 3: **Ablation on the design of FlashPruner**. We present results with all tokens as a reference. Experimental results on image (POPE) and video benchmarks (MVbench and EgoSchema) demonstrate that our model maintains robust performance even under extreme token compression, effectively retaining only 5% and 10% of tokens, respectively.

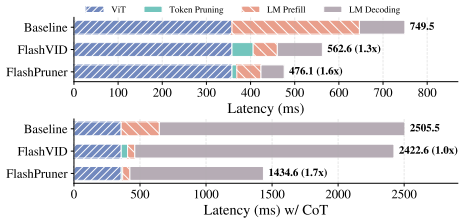


Figure 4: **Latency breakdown** for baseline (no token pruning), FlashVID and FlashPruner on MVBench under 10% vision tokens retained. Detailed numbers are included in Tab. 8.

When $p'_\theta = p_E$, the loss directly trains the early-exit draft distribution that proposes speculative tokens. When $p'_\theta = p_T$, the loss adapts the full-depth model to pruned vision tokens, preserving target quality after pruning. Since both branches are aligned to the same full-token teacher on the same on-policy prefixes, LoFT optimizes the distributional condition suggested by Theorem 1 while also adapting FlashPruner to the language model under pruned vision inputs.

During training, to train FlashPruner, we use a straight-through estimator [5] and directly copy gradients from the language model p_θ. We find that this simple strategy already works well for the initial experiments in this work. Specifically, we denote the hard selection mask $m_i^{\text{hard}} = \mathbb{I}[i \in \mathcal{I}_k]$ and $m = m_i^{\text{hard}} + \mathbf{s} - \text{sg}(\mathbf{s})$, where sg denotes the stop-gradient operation. During the forward pass, m is equivalent to m^{hard}, preserving the hard Top-k selection behavior. During the backward pass, the surrogate gradient allows gradients to be propagated to the token importance predictor h_φ through the discrete selection step.

4 Experiments

4.1 Settings

Benchmarks and metrics. We evaluate our method on multiple image and video understanding tasks. For image understanding, we choose GQA [20], MMBench [30], MME [16], POPE [26], ScienceQA [32], VQA$^{\text{Text}}$ [41], and OCRBench [31]. As for video understanding, we adopt four widely used video understanding benchmarks, including VideoMME [17], EgoSchema [34], LongVideoBench [51], and MVBench [25]. See Sec. B for more details. We report performance under different token retention ratios and relative accuracy (Rel. Acc.) compared with full tokens to assess the relative performance degradation under token pruning.

Models. We mainly evaluate our method on Qwen2.5-VL-7B [4] in the main text. For image understanding tasks, we include LLaVA-1.5-7B [28] as an additional comparison following previous work. For video understanding tasks, we also add LLaVAOneVision [23] and LLaVA-Video [63]. See Sec. E in the Appendix for more results. The above models share a typical "ViT-Projector-LM" architecture. We further discuss extensions to the Qwen3-VL [3] series with DeepStack [35] and Qwen3.5-family[1] models in Sec. G in the Appendix.

Implementation details. We set pooling size $b = 32$ (c.f. Eq. (1)) and window size $w = 16$ (c.f. Eq. (2)) in FlashPruner. And set the number of exit layers E to 25% of the number of language model layers, e.g., $E = 7$ for Qwen2.5-VL-7B. During inference, we set the number of draft tokens to $\gamma = 6$. During evaluation, we keep the same settings as baselines, such as the base model, number of retained tokens, and evaluation protocol, and report official results for fair comparison. See more evaluation details in Sec. D and training details in Sec. C.

[1] https://huggingface.co/collections/Qwen/qwen35

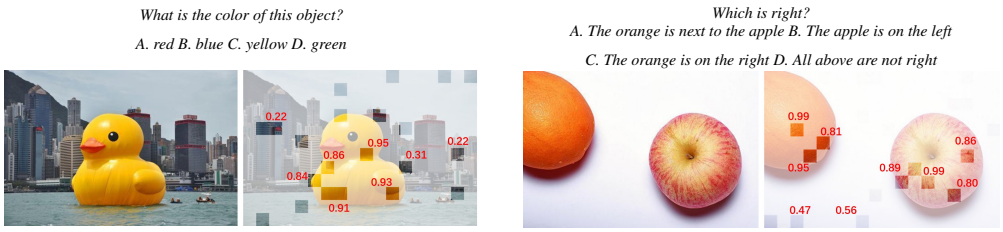


Figure 5: **Visualization of selected tokens**. We highlight importance score (sigmoid with s) with red and make the unselected tokens slightly transparent for clarity.

4.2 Main Results

Results on image understanding benchmarks. Tab. 1 shows that FlashPruner consistently outperforms prior token reduction methods across all evaluated image understanding benchmarks with different token retention budgets. When retaining 20% of vision tokens, FlashPruner achieves an average relative accuracy of 98.7%, substantially closing the gap to the full-token baseline and surpassing VisionZip, DivPrune, and MMTok by clear margins. Notably, FlashPruner not only maintains performance on standard VQA-style datasets such as GQA and VQA^{Text}, but also produces fewer object hallucinations compared to other token pruning approaches, as shown by results on POPE benchmark for hallucination evaluation.

As the token budget becomes more constrained, the advantage of FlashPruner becomes increasingly pronounced. Under the 10% setting, FlashPruner maintains 93.5% relative accuracy, outperforming the closest competitor (MMTok) by 5.0 points, while achieving better performance across all individual datasets. This trend continues in the extreme 5% regime, where FlashPruner still retains 82.2% of the full performance, significantly higher than MMTok (79.0%) and other baselines. Overall, these results demonstrate that FlashPruner maintains strong image understanding capabilities even with highly compressed vision tokens.

Results on video understanding benchmarks. Tab. 2 shows that FlashPruner consistently outperforms existing token reduction methods across diverse video understanding benchmarks under different token retention ratios. When retaining 25% of tokens, FlashPruner achieves an average relative accuracy of 99.8%, nearly matching the full-token baseline, and surpassing all prior methods. As the token budget decreases, FlashPruner continues to demonstrate superior performance. Under the 20% setting, FlashPruner achieves 98.9% relative accuracy, outperforming FlashVID, the strongest baseline, by a clear margin. This advantage remains consistent at 15% and 10% budgets, where FlashPruner achieves 98.0% and 96.4% relative accuracy, respectively. Compared to other methods, FlashPruner exhibits more stable performance degradation as token budgets shrink, especially on challenging long video benchmarks such as VideoMME (Long) and LongVideoBench, suggesting that FlashPruner effectively compresses long vision sequences. Results demonstrate that FlashPruner also works in the video understanding scenario, especially for long videos.

4.3 Analysis

Design of FlashPruner. Tab. 3 shows the effect of different components in FlashPruner. We find that the individual components generally contribute to performance improvements across all three tasks. However, the relative importance of each component varies across different tasks. We speculate that each task may possess an inherent bias regarding which pattern matters. Combining all modules consistently achieves the best performance across different tasks.

Latency analysis. Next, we analyze the efficiency of the proposed method. We report latency on a single NVIDIA RTX5090 GPU. Results shown in Fig. 4 show the time cost of different baselines on MVBench [25] with only 10% of vision tokens retained. We observe that our FlashPruner shows high efficiency for token selection compared with FlashVID due to our design being compatible with efficient attention implementations. By reducing the number of vision tokens, the time-to-first-token is largely reduced by 1.53× (646.5 ms *vs.* 423.6 ms). In the decoding phase, FlashPruner accelerates decoding by 1.96× compared with the baseline and generates 116.7 tokens per second. The total latency of prefill and decoding is reduced by 1.6× compared with the baseline. Furthermore, our FlashPruner achieves high speedup in the decoding stage, which substantially reduces the decoding

time. This advantage will be amplified when using CoT prompts because CoT increases decoding length greatly, as shown in Fig. 4. FlashPruner achieves 1.7× speedup while FlashVID has almost no speedup, which shows the importance of acceleration in the decoding stage.

Visualization. We visualize selected tokens in Fig. 5, showing that FlashPruner selects question-relevant vision tokens while preserving visual information.

5 Conclusion

In this paper, we present FlashPruner, an efficient and effective approach for accelerating inference of VLMs. FlashPruner selects the most important vision tokens effectively and accelerates the prefill stage. In addition, we propose LoFT to make self-speculative decoding with early layers accessible during the decoding stage to accelerate decoding. Extensive ablation and comparative experiments show that FlashPruner achieves high performance in terms of accuracy and speedup.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the Yangcheng Scholars Research Project (No.2024312049) and Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things(No.2023B1212010007). The authors declare no competing interests.

References

[1] Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, 2025.

[2] Changwoo Baek, Jouwon Song, Sohyeon Kim, and Kyeongbo Kong. Agilepruner: An empirical study of attention and diversity for adaptive visual token pruning in large vision-language models. In *ICLR*, 2026.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[7] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *ICML*, 2024.

[8] Junjie Chen, Xuyang Liu, Zichen Wen, Yiyu Wang, Siteng Huang, and Honggang Chen. Variation-aware vision token dropping for faster large vision-language models. In *CVPR*, 2026.

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

[10] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *NeurIPS*, 2024.

[11] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022.

[12] Sixun Dong, Juhua Hu, Mian Zhang, Ming Yin, Yanjie Fu, and Qi Qian. MMTok: Multimodal coverage maximization for efficient inference of VLMs. In *ICLR*, 2026.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[14] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *ACL*, 2024.

[15] Ziyang Fan, Keyu Chen, Ruilong Xing, Yulin Li, Li Jiang, and Zhuotao Tian. FlashVID: Efficient video large language models via training-free tree-based spatiotemporal token merging. In *ICLR*, 2026.

[16] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. MME: A comprehensive evaluation benchmark for multimodal large language models. In *NeurIPS*, 2025.

[17] 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. In *CVPR*, 2025.

[18] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[19] Xiaohu Huang, Hao Zhou, and Kai Han. PruneVid: Visual token pruning for efficient video large language models. In *ACL*, 2025.

[20] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019.

[21] Fengqing Jiang, Fengbo Ma, Zhangchen Xu, Yuetai Li, Zixin Rao, Bhaskar Ramasubramanian, Luyao Niu, Bo Li, Xianyan Chen, Zhen Xiang, and Radha Poovendran. SOSBENCH: Benchmarking safety alignment on scientific knowledge. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025.

[22] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

[23] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-onevision: Easy visual task transfer. *Transactions on Machine Learning Research*, 2025.

[24] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

[25] 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. In *CVPR*, 2024.

[26] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *EMNLP*, 2023.

[27] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *ICML*, 2024.

[28] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024.

[29] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[30] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, 2024.

[31] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67, 2023.

[32] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

[33] Fengbo Ma, Zixin Rao, Xiaoting Li, Zhetao Chen, Hongyue Sun, Yiping Zhao, Xianyan Chen, and Zhen Xiang. IntrAgent: An LLM agent for content-grounded information retrieval through literature review. In *ACL*, 2026.

[34] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *NeurIPS*, 2023.

[35] Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for LMMs. In *NeurIPS*, 2024.

[36] Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. *AI Open*, 6:314–322, 2025.

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[38] Zixin Rao, Youssef Mohamed, Shang Liu, and Zeyan Liu. Two birds with one stone: Multi-task detection and attribution of llm-generated text. In Wei Liang, Sun-Yuan Kung, and Meikang Qiu, editors, *Security and Privacy in Communication Networks*, 2026.

[39] Zixin Rao, Wentian Zhu, Chan Aristella Lu, Zhaorun Chen, Wei Niu, Le Guan, Bo Li, and Zhen Xiang. Fragfuse: Bypassing access control of large language model agents via memory-based query fragmentation and fusion. *arXiv preprint arXiv:2606.15609*, 2026.

[40] Zhenwei Shao, Mingyang Wang, Zhou Yu, Wenwen Pan, Yan Yang, Tao Wei, Hongyuan Zhang, Ning Mao, Wei Chen, and Jun Yu. Growing a twig to accelerate large vision-language models. In *ICCV*, 2025.

[41] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[43] Shuai Wang, Zhenhua Liu, Jiaheng Wei, Xuanwu Yin, Dong Li, and Emad Barsoum. Athena: Enhancing multimodal reasoning with data-efficient process reward models. *Transactions on Machine Learning Research*, 2026.

[44] Shuai Wang, Daoan Zhang, Tianyi Bai, Shitong Shao, Jiebo Luo, and Jiaheng Wei. Last: Learning to think in space and time for generalist vision-language models. *arXiv preprint arXiv:2511.19261*, 2025.

[45] Shuai Wang, Daoan Zhang, Zhe Tang, Hao Cheng, and Jiaheng Wei. Self-boosting vision-language models with noisy student on-policy self-distillation. *arXiv preprint arXiv:2607.23125*, 2026.

[46] Tiannan Wang, Jiamin Chen, Qingrui Jia, Shuai Wang, Ruoyu Fang, Huilin Wang, Zhaowei Gao, Chunzhao Xie, Chuou Xu, Jihong Dai, et al. Weaver: Foundation models for creative writing. *arXiv preprint arXiv:2401.17268*, 2024.

[47] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[48] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[49] Zichen Wen, Shaobo Wang, Yufa Zhou, Junyuan Zhang, Qintong Zhang, Yifeng Gao, Zhaorun Chen, Bin Wang, Weijia Li, Conghui He, and Linfeng Zhang. Efficient multi-modal large language models via progressive consistency distillation. In *NeurIPS*, 2025.

[50] Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, Rui Li, Thibaud Frere, Leandro von Werra, Aritra Roy Gosthipaty, and Andrés Marafioti. Finevision: Open data is all you need. *arXiv preprint arXiv:2510.17269*, 2025.

[51] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. In *NeurIPS*, 2024.

[52] Jingqi Xu, Jingxi Lu, Chenghao Li, Sreetama Sarkar, and Peter A. Beerel. Hivtp: A training-free method to improve vlms efficiency via hierarchical visual token pruning using middle-layer-based importance score. *arXiv preprint arXiv:2509.23663*, 2025.

[53] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[54] Jinrui Yang, Xianhang Li, Druv Pai, Yuyin Zhou, Yi Ma, Yaodong Yu, and Cihang Xie. Scaling white-box transformers for vision. In *NeurIPS*, 2024.

[55] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *CVPR*, 2025.

[56] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *ICLR*, 2025.

[57] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *CVPR*, 2024.

[58] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *ICCV*, 2025.

[59] Yu Zhang, Jinlong Ma, Yongshuai Hou, Xuefeng Bai, Kehai Chen, Yang Xiang, Jun Yu, and Min Zhang. Evaluating and steering modality preferences in multi-modal LLMs. In *ICML*, 2026.

[60] Yu Zhang, Chuyang Sun, Kehai Chen, Xuefeng Bai, Yang Xiang, and Min Zhang. Mitigating multimodal hallucination via phase-wise self-reward. *arXiv preprint arXiv:2604.17982*, 2026.

[61] Yu Zhang, Mufan Xu, Xuefeng Bai, Kehai Chen, Pengfei Zhang, Yang Xiang, and Min Zhang. Instruction anchor: Dissecting the mechanistic dynamics of modality arbitration. *arXiv preprint arXiv:2602.03677*, 2026.

[62] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *ICML*, 2025.

[63] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun MA, Ziwei Liu, and Chunyuan Li. LLaVA-video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research*, 2025.

[64] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *OSDI*, 2024.

[65] Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, et al. Agents: An open-source framework for autonomous language agents. *arXiv preprint arXiv:2309.07870*, 2023.

A Related Work

Vision-language models. The development of large language models [6, 39, 53, 38, 21, 46] and their agents [33, 65, 36] has reshaped the landscape of artificial intelligence. Meanwhile, VLMs have been growing based on powerful language models to perform visual understanding and reasoning. VLMs integrate a vision encoder (e.g., ViT [13, 54]), a projection module, and a language model [6] to perform multimodal tasks [4, 43, 44, 45, 47, 59, 60, 61]. Modern VLMs can handle high-resolution images [4] and long videos [63]. However, high-resolution images and long videos are encoded into multiple vision tokens by ViT, which introduces high complexity, especially for the attention module, and increases the prefill latency. In addition, for complex problems such as visual reasoning, VLMs usually generate multiple text tokens in the decoding phase to perform chain-of-thought reasoning [3, 48], which introduces high latency in the decoding stage.

Inference acceleration of VLMs. Previous methods for inference acceleration of VLMs mainly focus on token reduction utilizing visual token redundancy. FastV [9] first studies the redundancy among vision tokens and shows that pruning vision tokens after the vision encoder could accelerate inference of VLMs while maintaining the performance. Following it, many approaches are proposed to reduce the number of vision tokens using vision redundancy [9, 55, 58], vision diversity-based criterion [1, 12], text-to-vision relevance [62], trainable token selection modules [40], and consistency distillation [47, 49]. Many approaches usually use attention maps [9, 15, 40, 55, 52] as signals. This leads to two limitations: a) it is not compatible with FlashAttention [11] and leads to less inference speedup compared with theoretical gains and b) attention maps also suffer from internal bias in retaining tokens at the end of visual sequences according to [8]. In addition, although reducing vision tokens effectively accelerates the prefill stage of the language model part of VLMs, their speedup in the decoding stage is limited. To tackle the above limitations, we present FlashPruner, a lightweight vision token pruning block to accelerate the prefill stage and propose LoFT to adopt FlashPruner and self-speculative decoding to accelerate decoding during inference.

B Evaluation Benchmarks

In this section, we provide a brief overview of all benchmarks used in our experiments, including image and video tasks.

B.1 Image Benchmarks

Evaluation on image benchmarks in total involves 10 benchmarks as we present below where 7 benchmarks are used in the main text and 3 additional benchmarks are used in the appendix.

GQA [20] focuses on visual understanding and reasoning with spatial relationships and object properties. It evaluates VLMs' ability to perform visual reasoning under visual environments. For GQA, accuracy over all questions is reported.

MMBench [30] contains over 3000 multiple-choice questions spanning 20 ability dimensions to conduct object evaluation for VLMs. We report accuracy for validation set of MMBench with 4329 questions.

MME [16] evaluates VLMs in terms of perception and cognition abilities on a total of 14 sub-tasks. We report the summation of perception and cognition scores (P+C in Tabs. 1 and 5).

POPE [26] evaluates object hallucination by asking binary questions about objects in the image. We report F1 score following previous work [12].

VQA$^{\text{Text}}$ [41] evaluates a model's ability to analyze and reason about text in images. We report accuracy on the validation set with 5000 questions.

ScienceQA [32] (SQA) is a benchmark including 21K multimodal multiple-choice questions about natural, language, and social science topics. It evaluates VLMs' capability of multimodal comprehension, multi-step reasoning, and interpretability. We test on the test set for questions with images as input, which contains 2017 questions.

OCRBench [31] is designed to evaluate the optical character recognition capabilities of VLMs across 29 sub-tasks with five major categories—text recognition, scene text-centric VQA, document-oriented

VQA, key information extraction, and handwritten mathematical expression recognition. We report accuracy on 1000 questions.

VQA-V2 [18] provides a large-scale evaluation of VLMs about understanding of vision, language and commonsense knowledge. We report accuracy on the validation set with 214354 questions.

MMMU [57] is designed to test the ability of VLMs on massive multi-discipline tasks that require college-level subject knowledge and deliberate reasoning. We report accuracy on the validation set containing 900 questions.

SEED [24] consists of 19K multiple-choice questions with 12 evaluation domains. We report accuracy on 18K questions with images input.

B.2 Video Benchmarks

We evaluate our method using four benchmarks: VideoMME [17], EgoSchema [34], LongVideoBench [51] and MVBench [25].

VideoMME [17] includes 900 videos from six domains with 30 subfields. The duration of videos ranges from 11 seconds to one hour. It contains 2700 multiple-choice questions. We report the accuracy of all questions.

EgoSchema [34] has 5031 five-way choice questions and all videos are three minutes long. We report accuracy for the subset of 500 questions and the total 5031 questions.

LongVideoBench [51] is designed to evaluate the ability of VLMs to understand long videos. It includes 1337 questions from 752 videos for the validation set and 5341 questions from 3011 videos for the test set. We report results on the validation set.

MVBench [25] has 4000 questions from 20 task types. We report accuracy on all questions.

C Training Details

Training data. For Qwen2.5-VL-7B, since its post-training data is not publicly accessible, we use a subset from FineVision [50]. We randomly sample approximately 1M samples for Stage 1 and 6K samples for Stage 2. To support efficient inference for videos, we also randomly sample a portion of LLaVA-Video-178K [63] in two stages, with 100K samples for Stage 1 and 1K for Stage 2.

For experiments with LLaVA series models, we use the corresponding post-training dataset for each model.

Training data and details. We summarize training hyperparameters in Tab. 4.

D Evaluation Details

For Qwen2.5-VL-7B, we set the minimum and maximum number of pixels per image to $256 \times 28 \times 28$ and $2048 \times 28 \times 28$ on image benchmarks following [12], respectively. For video benchmarks, we sample 32 frames following [15]. We use greedy decoding for stable and reproducible evaluation and follow settings in `lmms-eval`.[2]

E More Experimental Results and Analysis

E.1 More Evaluation on Other Models

We validate our method with other series models to show the generalization of the proposed method.

Results on LLaVA-1.5-7B. We compare our method with previous works on LLaVA-1.5-7B [28], a popular model used in previous work. For benchmarks, we add VQA-v2 [18], MMMU [57], SEED [24] following [12]. We add two additional baselines: TwigVLM [40], a training-based approach, and SparseVLM [62].

[2] https://github.com/evolvinglmms-lab/lmms-eval

We list results in Tab. 5. Different from Qwen2.5-VL-7B, LLaVA-1.5-7B has a fixed number of vision tokens for arbitrary visual inputs, i.e., 576 tokens for every image. FlashPruner achieves the best performance under different retained token rates. Specifically, FlashPruner gets 100.1%, 99.6%, and 98.3% with 192/128/64 vision tokens.

Results on LLaVAOneVision and LLaVA-Video. To extend evaluation to video understanding tasks, we further evaluate FlashPruner with LLaVAOneVision(-7B) and LLaVA-Video(-7B). We add PruneVID [19] following [15]. We sample 32 and 64 frames for LLaVAOneVision and LLaVA-Video, respectively.

Results are shown in Tab. 6 and Tab. 7. Notably, FlashPruner consistently outperforms previous state-of-the-art methods by a large margin across different retained token ratios and benchmarks.

E.2 More Ablation Study

Hyperparameters for self-speculative decoding. We first study the effect of different hyperparameters for self-speculative decoding, i.e., the number of draft tokens γ and the exit layer E. Fig. 6 shows the number of generated tokens per second under different settings in the decoding stage. We find that decoding speed generally improves across different exit layers and draft token counts compared to the autoregressive baseline. For a given number of draft tokens, decoding speed increases as E decreases. We observe that $\gamma = 6$ achieves the highest speedup ($\sim 1.96\times$). We also study the acceptance rate of draft tokens, as shown in Fig. 6. Fig. 6 indicates that an excessively large γ generates too many tokens that fail verification, which wastes the draft model's computation and ultimately reduces the overall speedup.

Latency breakdown. We present detailed statistics of Fig. 4 in Tab. 8.

Ablation on training methods. We compare different training strategies in Fig. 7. We observe that the warm-up stage improves uniform sampling at all ratios of retained tokens, which shows the effectiveness of FlashPruner. After the warm-up stage, we try three different strategies in the second stage: SFT, off-policy KD and on-policy distillation. Off-policy KD and SFT achieve comparable performance, while on-policy distillation outperforms them clearly, especially in lower token-retention ratios, e.g., 10% and 15% retained tokens.

E.3 Efficiency Analysis.

Efficiency analysis of FlashPruner. FlashPruner introduces additional parameters $9d^2 + 3d + 3$. For example, for Qwen2.5-VL-7B [4], $d = 5120$ and FlashPruner has 236M parameters, which contains only 3% of the total parameters of Qwen2.5-VL-7B. Moreover, FlashPruner is highly efficient with FlashAttention because we only need the final output and do not need the attention map.

In addition, when we select vision tokens based on the score $\mathbf{s}$, we only need the set of selected tokens and do not need to sort them. In practice, for long sequences, we use:

```
flashinfer.top_k(s, k, sorted=False)
```

and for short sequences, we use:

```
torch.topk(s, k, sorted=False)
```

to get the index set $\mathcal{I}_K$ efficiently[3].

Additional cache analysis in self-speculative decoding. To improve efficiency in self-speculative decoding, we save the query vectors in the E-th layer. Here we analyze the additional cache compared with standard KV size in autoregressive models. Consider a model with group query attention with g KV heads, h heads, T layers, current token length N, hidden dimension d, we have dimension per head $d_h = d/h$. The total KV cache is $2TNd_hg$, and the additional Q cache is hd_h, so the additional cache ratio is $\frac{hd_h}{2TNd_hg} = \frac{h}{2TNg}$. For Qwen2.5-VL-7B, $h = 28$, $g = 4$, $T = 28$, so the ratio is $\frac{1}{8N}$. For most situations, $N > 100$ only requires $< 0.125\%$ additional cache, which is negligible.

[3] https://docs.flashinfer.ai/api/topk.html

F Proof of Theorem 1

In this section, we restate Theorem 1 as the more detailed Theorem 1 and prove it.

Let $\mathcal{V}$ be a finite next-token vocabulary, let $\Delta(\mathcal{V})$ be the probability simplex over $\mathcal{V}$, and let all logarithms be natural. For $u, v \in \Delta(\mathcal{V})$, write $u \ll v$ when $v(x) = 0$ implies $u(x) = 0$. Define

$$\mathrm{KL}(u\|v) := \begin{cases} \sum_{x:u(x)>0} u(x) \log \dfrac{u(x)}{v(x)}, & u \ll v, \\ +\infty, & \text{otherwise}, \end{cases} \tag{11}$$

and

$$\mathrm{TV}(u, v) := \frac{1}{2} \sum_{x \in \mathcal{V}} |u(x) - v(x)|. \tag{12}$$

The distributions below are the post-processing distributions used by the sampler. Whenever $q_c, p_c \in \Delta(\mathcal{V})$ are a draft/target pair, for x with $q_c(x) > 0$, define

$$a_c(x) = \min\left\{1, \frac{p_c(x)}{q_c(x)}\right\}. \tag{13}$$

The value of $a_c(x)$ on $\{x : q_c(x) = 0\}$ is arbitrary. Define

$$\alpha(c) = \mathbb{E}_{X \sim q_c}[a_c(X)], \qquad \bar{\alpha}_\rho = \mathbb{E}_{c \sim \rho} \alpha(c). \tag{14}$$

Assume all context-indexed probability kernels appearing below are measurable. The bounds concern stochastic speculative acceptance, not wall-clock speedup or the output law of a full sampler. The rollout-to-inference transfer conditions are assumptions, not consequences of the LoFT objective alone.

Theorem 1 (Conditional LoFT-transfer-to-acceptance bound)**.** *Let $\mathcal{C}$ be a measurable paired-context space. For each $c = (s, \pi(s)) \in \mathcal{C}$ with a shared generated prefix, let $r_c = r(\cdot \mid s)$ be the full-token teacher distribution, $q_c = q(\cdot \mid \pi(s))$ the early-exit draft distribution, and $p_c = p(\cdot \mid \pi(s))$ the full-depth target distribution, all in $\Delta(\mathcal{V})$. Let ν be the inference-time paired-context distribution. If, for finite $\varepsilon_E, \varepsilon_T \geq 0$,*

$$\mathbb{E}_\nu \mathrm{KL}(q_c\|r_c) \leq \varepsilon_E, \qquad \mathbb{E}_\nu \mathrm{KL}(p_c\|r_c) \leq \varepsilon_T, \tag{15}$$

then

$$\bar{\alpha}_\nu \geq 1 - \sqrt{\frac{\varepsilon_E}{2}} - \sqrt{\frac{\varepsilon_T}{2}}. \tag{16}$$

If, for a LoFT rollout distribution μ on $\mathcal{C}$,

$$\mathcal{L}_E = \mathbb{E}_\mu \mathrm{KL}(q_c\|r_c), \quad \mathcal{L}_T = \mathbb{E}_\mu \mathrm{KL}(p_c\|r_c), \quad \mathcal{L}_{\mathrm{LoFT}} = \lambda \mathcal{L}_E + (1-\lambda)\mathcal{L}_T \leq \varepsilon, \tag{17}$$

with $\lambda \in (0, 1)$, $0 \leq \varepsilon < \infty$, and $\beta_E, \beta_T, \xi_E, \xi_T \geq 0$, and

$$\mathbb{E}_\nu \mathrm{KL}(q_c\|r_c) \leq \beta_E \mathcal{L}_E + \xi_E, \qquad \mathbb{E}_\nu \mathrm{KL}(p_c\|r_c) \leq \beta_T \mathcal{L}_T + \xi_T, \tag{18}$$

then

$$\bar{\alpha}_\nu \geq 1 - \sqrt{\frac{\beta_E \varepsilon/\lambda + \xi_E}{2}} - \sqrt{\frac{\beta_T \varepsilon/(1-\lambda) + \xi_T}{2}}. \tag{19}$$

Finally, for a block of $\gamma \in \mathbb{N}_+$ draft tokens, let A_γ be the accepted prefix length. Suppose there is a filtration $\{\mathcal{F}_i\}_{i=0}^\gamma$. For every $i = 1, \ldots, \gamma$, suppose $\{A_\gamma \geq i-1\} \in \mathcal{F}_{i-1}$, C_i and X_i are globally defined random variables, and on the active event $\{A_\gamma \geq i-1\}$, C_i is $\mathcal{F}_{i-1}$-measurable and $X_i \mid \mathcal{F}_{i-1} \sim q_{C_i}$; on the complementary stopped event they are arbitrary. Assume also that, for every $i = 1, \ldots, \gamma$,

$$\mathbb{E}\big[\mathbf{1}_{\{A_\gamma \geq i\}} \mid \sigma(\mathcal{F}_{i-1}, X_i)\big] = \mathbf{1}_{\{A_\gamma \geq i-1\}} a_{C_i}(X_i). \tag{20}$$

If, for some finite $\varepsilon_{\mathrm{pos}} \geq 0$,

$$\mathrm{KL}(q_{C_i}\|p_{C_i}) \leq \varepsilon_{\mathrm{pos}} \quad \text{a.s. on } \{A_\gamma \geq i-1\}, \qquad i = 1, \ldots, \gamma, \tag{21}$$

then with $\alpha_{\mathrm{pos}} = 1 - \sqrt{\varepsilon_{\mathrm{pos}}/2}$, if $0 \leq \alpha_{\mathrm{pos}} \leq 1$,

$$\mathbb{E}[A_\gamma] \geq \sum_{i=1}^{\gamma} \alpha_{\mathrm{pos}}^i. \tag{22}$$

If $0 \le \alpha_{\text{pos}} < 1$, *then*

$$\mathbb{E}[A_\gamma] \ge \frac{\alpha_{\text{pos}}(1-\alpha_{\text{pos}}^\gamma)}{1-\alpha_{\text{pos}}}. \tag{23}$$

Moreover, for all $\varepsilon_{\text{pos}} \ge 0$,

$$\frac{\mathbb{E}[A_\gamma]}{\gamma} \ge \max\left\{0,\, 1 - \frac{\gamma+1}{2}\sqrt{\frac{\varepsilon_{\text{pos}}}{2}}\right\}. \tag{24}$$

To prove our theorem, we first make the following assumptions.

Assumption 1 (Post-processed stochastic speculative rule)**.** *For any context* c, $q_c, p_c \in \Delta(\mathcal{V})$ *are the distributions after all decoding transforms. For* x *with* $q_c(x) > 0$,

$$a_c(x) = \min\left\{1, \frac{p_c(x)}{q_c(x)}\right\}. \tag{25}$$

The value of $a_c(x)$ *on* $\{x : q_c(x) = 0\}$ *is arbitrary.*

Assumption 2 (Paired teacher-bridge context)**.** *There is a pruning map* $\pi : \mathcal{S} \to \mathcal{S}_{\text{pr}}$. *Each paired state is* $c = (s, \pi(s))$ *with a shared generated prefix. The distributions* $r_c = r(\cdot \mid s)$, $q_c = q(\cdot \mid \pi(s))$, *and* $p_c = p(\cdot \mid \pi(s))$ *belong to* $\Delta(\mathcal{V})$.

Assumption 3 (Mixed LoFT transfer)**.** *The rollout distribution* μ *and inference distribution* ν *are defined on the same paired-context space. For* $\lambda \in (0,1)$,

$$\mathcal{L}_E = \mathbb{E}_\mu \,\mathrm{KL}(q_c \| r_c), \qquad \mathcal{L}_T = \mathbb{E}_\mu \,\mathrm{KL}(p_c \| r_c), \qquad \mathcal{L}_{\text{LoFT}} = \lambda\mathcal{L}_E + (1-\lambda)\mathcal{L}_T. \tag{26}$$

There exist constants $\beta_E, \beta_T, \xi_E, \xi_T \ge 0$ *such that*

$$\mathbb{E}_\nu \,\mathrm{KL}(q_c \| r_c) \le \beta_E \mathcal{L}_E + \xi_E, \qquad \mathbb{E}_\nu \,\mathrm{KL}(p_c \| r_c) \le \beta_T \mathcal{L}_T + \xi_T. \tag{27}$$

Assumption 4 (Sequential block process)**.** *Let* $\gamma \in \mathbb{N}_+$, $A_\gamma \in \{0, \dots, \gamma\}$, *and* $\{\mathcal{F}_i\}_{i=0}^\gamma$ *be the filtration generated by the accepted prefix, draft tokens, and accept/reject decisions up to position* i. *For each* $i = 1, \dots, \gamma$, $\{A_\gamma \ge i-1\} \in \mathcal{F}_{i-1}$, *and* C_i, X_i *are globally defined random variables. On* $\{A_\gamma \ge i-1\}$, C_i *is* $\mathcal{F}_{i-1}$*-measurable and* $X_i \mid \mathcal{F}_{i-1} \sim q_{C_i}$. *On* $\{A_\gamma < i-1\}$, C_i *and* X_i *are arbitrary. Moreover, for each* $i = 1, \dots, \gamma$,

$$\mathbb{E}\big[\mathbf{1}_{\{A_\gamma \ge i\}} \mid \sigma(\mathcal{F}_{i-1}, X_i)\big] = \mathbf{1}_{\{A_\gamma \ge i-1\}} a_{C_i}(X_i). \tag{28}$$

Assumption 5 (Uniform adaptive positionwise KL)**.** *For each* $i \in \{1, \dots, \gamma\}$,

$$\mathrm{KL}(q_{C_i} \| p_{C_i}) \le \varepsilon_{\text{pos}} \quad \textit{a.s. on } \{A_\gamma \ge i-1\}. \tag{29}$$

Lemma 1 (Acceptance identity)**.** *Under Assumption 1,*

$$\alpha(c) = 1 - \mathrm{TV}(q_c, p_c). \tag{30}$$

Proof.

$$\begin{aligned}
\alpha(c) &= \sum_{x : q_c(x) > 0} q_c(x) \min\left\{1, \frac{p_c(x)}{q_c(x)}\right\} && (31)\\
&= \sum_{x : q_c(x) > 0} \min\{q_c(x), p_c(x)\} && (32)\\
&= \sum_{x \in \mathcal{V}} \min\{q_c(x), p_c(x)\} && (33)\\
&= \frac{1}{2} \sum_{x \in \mathcal{V}} \big(q_c(x) + p_c(x) - |q_c(x) - p_c(x)|\big) && (34)\\
&= 1 - \mathrm{TV}(q_c, p_c). && (35)
\end{aligned}$$

□

Lemma 2 (Inference-time KL acceptance)**.** *Under Assumption 1, let ν be a distribution over contexts. Suppose*

$$\mathbb{E}_\nu \operatorname{KL}(q_c \| p_c) \le \varepsilon_{\inf} < \infty. \tag{36}$$

Then

$$\mathbb{E}_\nu \alpha(c) \ge 1 - \sqrt{\frac{\varepsilon_{\inf}}{2}}. \tag{37}$$

Proof. By Lemma 1, Pinsker's inequality, and Jensen's inequality,

$$\begin{aligned}
\mathbb{E}_\nu \alpha(c) &= 1 - \mathbb{E}_\nu \operatorname{TV}(q_c, p_c) && (38)\\
&\ge 1 - \mathbb{E}_\nu \sqrt{\frac{1}{2} \operatorname{KL}(q_c \| p_c)} && (39)\\
&\ge 1 - \sqrt{\frac{1}{2} \mathbb{E}_\nu \operatorname{KL}(q_c \| p_c)} && (40)\\
&\ge 1 - \sqrt{\frac{\varepsilon_{\inf}}{2}}. && (41)
\end{aligned}$$

□

Lemma 3 (Teacher bridge with explicit tolerances)**.** *Under Assumptions 1 and 2, let ν be a distribution over paired contexts. Suppose that, for finite $\eta_E, \eta_T \ge 0$,*

$$\mathbb{E}_\nu \operatorname{KL}(q_c \| r_c) \le \eta_E, \qquad \mathbb{E}_\nu \operatorname{KL}(p_c \| r_c) \le \eta_T. \tag{42}$$

Then

$$\bar{\alpha}_\nu \ge 1 - \sqrt{\frac{\eta_E}{2}} - \sqrt{\frac{\eta_T}{2}}. \tag{43}$$

Proof. For ν-almost every c,

$$\operatorname{TV}(q_c, p_c) \le \operatorname{TV}(q_c, r_c) + \operatorname{TV}(p_c, r_c) \le \sqrt{\frac{\operatorname{KL}(q_c \| r_c)}{2}} + \sqrt{\frac{\operatorname{KL}(p_c \| r_c)}{2}}. \tag{44}$$

Using Lemma 1,

$$\begin{aligned}
\bar{\alpha}_\nu &= 1 - \mathbb{E}_\nu \operatorname{TV}(q_c, p_c) && (45)\\
&\ge 1 - \mathbb{E}_\nu \sqrt{\frac{\operatorname{KL}(q_c \| r_c)}{2}} - \mathbb{E}_\nu \sqrt{\frac{\operatorname{KL}(p_c \| r_c)}{2}} && (46)\\
&\ge 1 - \sqrt{\frac{\mathbb{E}_\nu \operatorname{KL}(q_c \| r_c)}{2}} - \sqrt{\frac{\mathbb{E}_\nu \operatorname{KL}(p_c \| r_c)}{2}} && (47)\\
&\ge 1 - \sqrt{\frac{\eta_E}{2}} - \sqrt{\frac{\eta_T}{2}}. && (48)
\end{aligned}$$

□

Lemma 4 (Mixed-loss relaxation)**.** *Under Assumptions 1, 2, and 3, if $0 \le \mathcal{L}_{\mathrm{LoFT}} \le \varepsilon < \infty$, then*

$$\bar{\alpha}_\nu \ge 1 - \sqrt{\frac{\beta_E \varepsilon/\lambda + \xi_E}{2}} - \sqrt{\frac{\beta_T \varepsilon/(1-\lambda) + \xi_T}{2}}. \tag{49}$$

Proof. Since $\mathcal{L}_E, \mathcal{L}_T \ge 0$,

$$\begin{aligned}
\lambda \mathcal{L}_E &\le \lambda \mathcal{L}_E + (1-\lambda)\mathcal{L}_T = \mathcal{L}_{\mathrm{LoFT}} \le \varepsilon, && (50)\\
(1-\lambda)\mathcal{L}_T &\le \lambda \mathcal{L}_E + (1-\lambda)\mathcal{L}_T = \mathcal{L}_{\mathrm{LoFT}} \le \varepsilon. && (51)
\end{aligned}$$

Since $\lambda \in (0,1)$, $\mathcal{L}_E \le \varepsilon/\lambda$ and $\mathcal{L}_T \le \varepsilon/(1-\lambda)$. Substituting these inequalities into (27) gives

$$\mathbb{E}_\nu \operatorname{KL}(q_c \| r_c) \le \beta_E \varepsilon/\lambda + \xi_E, \qquad \mathbb{E}_\nu \operatorname{KL}(p_c \| r_c) \le \beta_T \varepsilon/(1-\lambda) + \xi_T. \tag{52}$$

Apply Lemma 3 with $\eta_E = \beta_E \varepsilon/\lambda + \xi_E$ and $\eta_T = \beta_T \varepsilon/(1-\lambda) + \xi_T$. □

Lemma 5 (Accepted-prefix bound). *Under Assumptions 1, 4, and 5, let* $\delta = \sqrt{\varepsilon_{\text{pos}}/2}$ *and* $\alpha_{\text{pos}} = 1 - \delta$. *If* $0 \le \alpha_{\text{pos}} \le 1$, *then*

$$\mathbb{E}[A_\gamma] \ge \sum_{i=1}^{\gamma} \alpha_{\text{pos}}^i. \tag{53}$$

If $0 \le \alpha_{\text{pos}} < 1$, *then*

$$\mathbb{E}[A_\gamma] \ge \frac{\alpha_{\text{pos}}(1 - \alpha_{\text{pos}}^\gamma)}{1 - \alpha_{\text{pos}}}. \tag{54}$$

For all $\delta \ge 0$,

$$\frac{\mathbb{E}[A_\gamma]}{\gamma} \ge \max\left\{0,\, 1 - \frac{\gamma+1}{2}\delta\right\}. \tag{55}$$

Proof. Fix $i \in \{1, \dots, \gamma\}$. By Assumption 4 and the tower property,

$$\begin{aligned}
\mathbb{E}\big[\mathbf{1}_{\{A_\gamma \ge i\}} \mid \mathcal{F}_{i-1}\big] &= \mathbb{E}\big[\mathbb{E}\big[\mathbf{1}_{\{A_\gamma \ge i\}} \mid \sigma(\mathcal{F}_{i-1}, X_i)\big] \mid \mathcal{F}_{i-1}\big] && (56)\\
&= \mathbb{E}\big[\mathbf{1}_{\{A_\gamma \ge i-1\}} a_{C_i}(X_i) \mid \mathcal{F}_{i-1}\big] && (57)\\
&= \mathbf{1}_{\{A_\gamma \ge i-1\}} \mathbb{E}[a_{C_i}(X_i) \mid \mathcal{F}_{i-1}] && (58)\\
&= \mathbf{1}_{\{A_\gamma \ge i-1\}} \alpha(C_i). && (59)
\end{aligned}$$

Lemma 1, Pinsker's inequality, and Assumption 5 imply

$$\alpha(C_i) \ge \alpha_{\text{pos}} \quad \text{a.s. on } \{A_\gamma \ge i-1\}. \tag{60}$$

Therefore

$$\begin{aligned}
\Pr(A_\gamma \ge i) &= \mathbb{E}\big[\mathbb{E}\big[\mathbf{1}_{\{A_\gamma \ge i\}} \mid \mathcal{F}_{i-1}\big]\big] && (61)\\
&\ge \alpha_{\text{pos}} \Pr(A_\gamma \ge i-1). && (62)
\end{aligned}$$

Since $\Pr(A_\gamma \ge 0) = 1$, induction gives

$$\Pr(A_\gamma \ge i) \ge \alpha_{\text{pos}}^i, \qquad i = 1, \dots, \gamma. \tag{63}$$

Since $A_\gamma \in \{0, \dots, \gamma\}$,

$$\mathbb{E}[A_\gamma] = \sum_{i=1}^{\gamma} \Pr(A_\gamma \ge i) \ge \sum_{i=1}^{\gamma} \alpha_{\text{pos}}^i, \tag{64}$$

which gives (53). The finite geometric sum gives (54). For (55), if $0 \le \delta \le 1$, then Bernoulli's inequality gives $\big(1-\delta\big)^i \ge 1 - i\delta$ and

$$\frac{\mathbb{E}[A_\gamma]}{\gamma} \ge \frac{1}{\gamma}\sum_{i=1}^{\gamma}(1 - i\delta) = 1 - \frac{\gamma+1}{2}\delta. \tag{65}$$

For all $\delta \ge 0$, $\mathbb{E}[A_\gamma]/\gamma \ge 0$. Combining this nonnegativity with the preceding linear bound when $0 \le \delta \le 1$ proves (55); when $\delta > 1$, the maximum is 0 because $1 - (\gamma+1)\delta/2 < 0$. □

Proof of Theorem 1. The definitions of a_c, $\alpha(c)$, and the paired kernels instantiate Assumptions 1 and 2. Equation (16) follows from Lemma 3 with $\eta_E = \varepsilon_E$ and $\eta_T = \varepsilon_T$. The mixed hypotheses in (17)–(18) instantiate Assumption 3, so (19) follows from Lemma 4. The block hypotheses in (20)–(21) instantiate Assumptions 4 and 5, so (22), (23), and (24) follow from Lemma 5. □

Remark 1 (Scope). All results are conditional on the stated KL and transfer assumptions. The mixed LoFT bound is informative when β_E, β_T are moderate, ξ_E, ξ_T are small, and λ is bounded away from 0 and 1. Otherwise the inequalities remain valid but can become vacuous.

Table 4: Training hyperparameters.

	Stage 1	**Stage 2**
GPU	8 × NVIDIA A800	
Optimizer	AdamW	
Weight decay	0	
Optimizer parameters	$\beta_1 = 0.9, \beta_2 = 0.98$	
Batch size	128	64
Learning rate schedule	Cosine decay	Constant
Learning rate	$5e^{-5}$	$1e^{-6}$
Learning rate warm-up	Linear	-
Warm-up rate	0.03	-
# Epoch / iteration	1 epoch	200 steps
# Rollout	-	1
Sampling temperature	-	0.8
Top-p	-	0.95
Attention backend	FlashAttention-2	
Max generation length	-	2048
Update parameters	h_φ	h_φ, p_θ
Gradient Clip Norm	1.0	
Training backend	FSDP2	
Rollout backend	-	SGLang

G Discussion about Qwen3 and Qwen3.5

The proposed method FlashPruner could be used in general VLMs that follow the "ViT-Projector-LM" architecture. Here are two models that need to be discussed.

The first is the Qwen3-VL series model [3]. Qwen3-VL models still follow the typical "ViT-Projector-LM" architecture but use DeepStack [35] that merges features from different layers of ViT into the first layers of the language model. Here we simply use the same selected token indices for features of early ViT layers and this strategy does not increase additional computation or memory required. We found that our simple strategy could work well and maintain accuracy. We use Qwen3-VL-8B(-Instruct) [3] and evaluate it with 10% and 20% vision tokens on MMMU [57], MMBench(-dev-en-v1.1) [30], RealWorldQA [4], and MMstar [10]. We follow the official guide to set top-k to 20, top-p to 0.8, temperature to 0.7, presence-penalty to 1.5 and output length to 16384 for sampling tokens and report average results on five runs. Results are shown in Tab. 9. It shows that FlashPruner still works with VLMs with DeepStack [35] even with the simplest strategy. As for exploring other strategies, we leave it for future work.

For Qwen3.5, it is not trivial to adopt it for Qwen3.5 models with linear attention such as Gated DeltaNet [56] for self-speculative decoding. To integrate speculative decoding into linear attention models is out of the scope of this paper and we leave it for future research.

H Limitations and Social Impact

H.1 Limitations.

While FlashPruner and LoFT are effective at accelerating inference of VLMs, they require training models with additional computation. However, only one training is required. After that, inference acceleration could benefit models all the time. In addition, on some tasks, such as OCR, reducing the number of vision tokens leads to unignorable performance degradation. While effective with self-speculative decoding, it is non-trivial to adopt it to models with linear attention. Finally, due to the large computation cost, we did not report error bars because it would be too computationally expensive.

[4] https://huggingface.co/datasets/xai-org/RealworldQA

Table 5: Performance Comparison on LLaVA-1.5-7B. Results of other baselines are from MMTok [12] and TwigVLM [40].

Method	**GQA**	**MMB**	**MME**	**POPE**	**SQA**	**VQA^{V2}**	**VQA^{Text}**	**MMMU**	**SEED**	**Avg.**
Metric	Acc.	Acc.	P+C	F1	Acc.	Acc.	Acc.	Acc.	Acc.	Rel. Acc.(%)
Upper Bound, 576 Tokens										
LLaVA-1.5-7B	61.9	64.7	1862	85.9	69.5	78.5	58.2	36.3	58.6	100
Retain 192 Tokens ↓ 67%										
FastV	52.7	61.2	1612	64.8	67.3	67.1	52.5	34.3	57.1	89.6
SparseVLM	57.6	62.5	1721	83.6	69.1	75.6	56.1	33.8	55.8	95.5
VisionZip	59.3	63.0	1783	85.3	68.9	76.8	57.3	36.6	56.4	97.9
DivPrune	60.0	62.5	1762	87.0	68.7	76.9	57.0	35.4	58.7	98.0
VisionZip‡	60.1	63.4	1834	84.9	68.2	77.4	57.8	36.2	57.1	98.4
MMTok	60.1	63.4	1774	86.4	68.8	77.1	57.7	36.3	59.2	98.7
TwigVLM	61.2	64.0	1848	87.2	68.8	78.1	58.0	36.6	-	-
FlashPruner	61.7	64.2	1858	87.0	69.2	78.5	58.0	36.3	59.4	100.1
Retain 128 Tokens ↓ 78%										
FastV	49.6	56.1	1490	59.6	60.2	61.8	50.6	34.9	55.9	84.4
SparseVLM	56.0	60.0	1696	80.5	67.1	73.8	54.9	33.8	53.4	92.9
VisionZip	57.6	62.0	1762	83.2	68.9	75.6	56.8	37.9	54.9	96.8
DivPrune	59.3	62.0	1718	86.7	68.7	76.0	56.1	35.6	57.0	96.9
VisionZip‡	58.9	62.6	1823	83.7	68.3	76.6	57.0	37.3	55.8	97.7
MMTok	59.3	62.3	1779	86.3	68.8	76.4	57.0	35.7	58.6	97.8
TwigVLM	60.6	63.5	1818	86.6	69.5	77.9	57.8	36.6	-	-
FlashPruner	61.2	63.8	1840	87.0	69.0	78.2	58.0	36.1	59.0	99.6
Retain 64 Tokens ↓ 89%										
FastV	46.1	48.0	1256	48.0	51.1	55.0	47.8	34.0	51.9	75.6
SparseVLM	52.7	56.2	1505	75.1	62.2	68.2	51.8	32.7	51.1	86.9
VisionZip	55.1	60.1	1690	77.0	69.0	72.4	55.5	36.2	52.2	93.2
DivPrune	57.8	59.3	1674	85.6	68.1	74.1	54.7	35.6	55.1	94.8
VisionZip‡	57.0	61.5	1756	80.9	68.8	74.2	56.0	35.6	53.4	95.0
MMTok	58.3	61.2	1715	85.8	69.2	75.2	56.0	36.1	57.2	96.6
TwigVLM	58.8	60.4	1760	82.7	70.0	75.6	55.8	35.9	-	-
FlashPruner	59.4	62.8	1800	86.3	68.8	77.4	56.2	36.1	58.8	98.3

H.2 Social Impact

Positive impacts. We study the inference acceleration of VLMs, which makes large-scale foundation models more efficient and accessible in the real world.

Negative impacts. This paper does not inherently involve or encourage applications with direct negative societal impacts. We focus on a generic improvement in the field of machine learning, particularly in the inference acceleration of VLMs.

Table 6: Performance Comparison on LLaVAOneVision.

Method	VideoMME				EgoSchema		LongVideo Bench	MVBench	Avg.	
	Short	Medium	Long	Overall	Subset	Total			Acc.	Rel. Acc. (%)
100% Tokens										
LLaVAOneVision	69.9	56.7	48.9	58.5	62.2	60.3	56.6	58.3	58.4	100.0
Retain 25% Tokens										
FastV	68.1	54.7	46.8	56.5	60.4	57.8	55.4	56.4	56.5	96.7
VisionZip	68.8	57.3	48.2	58.1	63.0	60.4	56.4	57.8	58.2	99.7
PruneVID	67.3	54.8	47.2	56.4	61.0	58.1	55.4	56.8	56.7	97.1
FastVID	69.9	56.3	47.4	57.9	61.2	59.5	55.9	58.1	57.8	99.0
FlashVID	71.2	57.0	49.3	59.2	63.4	60.4	56.8	58.0	58.6	100.3
FlashPruner	72.4	58.0	51.2	60.5	64.6	62.0	58.5	59.6	60.2	103.1
Retain 20% Tokens										
FastV	66.3	53.9	46.9	55.7	60.6	57.6	56.0	56.0	56.3	96.4
VisionZip	68.6	57.0	48.3	58.0	62.0	60.0	55.4	57.6	57.7	98.8
PruneVID	67.2	53.9	48.2	56.4	63.2	60.2	55.2	56.2	57.0	97.6
FastVID	69.9	56.3	47.4	57.9	61.2	59.5	55.9	58.1	57.9	99.1
FlashVID	70.1	55.4	48.9	58.2	63.0	60.1	58.5	58.2	58.7	100.5
FlashPruner	72.4	58.1	51.0	60.5	63.8	63.8	60.2	59.4	61.0	104.5
Retain 15% Tokens										
FastV	64.6	54.0	45.3	54.6	59.8	56.6	54.8	55.0	55.2	94.5
VisionZip	63.8	54.6	48.3	55.6	62.8	60.0	54.1	53.5	55.8	95.5
PruneVID	67.2	52.8	46.7	56.1	61.6	57.7	54.5	55.1	55.7	95.4
FastVID	69.7	55.8	47.7	57.7	58.8	58.9	56.7	58.2	57.9	99.1
FlashVID	69.6	56.0	48.9	58.2	62.8	60.4	57.5	57.9	58.5	100.2
FlashPruner	71.0	57.2	50.9	59.7	63.4	62.5	59.5	58.8	60.1	102.9
Retain 10% Tokens										
FastV	60.9	52.2	44.9	52.7	59.0	56.0	52.4	53.4	53.6	91.8
VisionZip	60.3	52.9	46.7	53.3	61.6	58.5	49.4	54.8	54.0	92.5
PruneVID	65.9	52.8	45.6	54.7	60.0	57.2	54.0	53.7	54.9	94.0
FastVID	68.1	55.7	47.8	57.2	58.8	58.7	55.7	57.0	57.1	97.8
FlashVID	67.3	57.1	49.0	57.8	62.4	60.0	56.5	57.4	57.9	99.1
FlashPruner	69.1	56.8	50.2	58.7	63.0	61.8	58.4	58.3	59.3	101.5

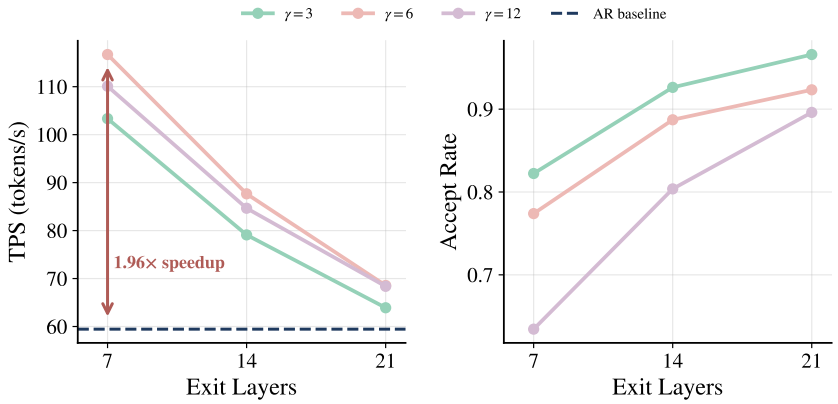


Figure 6: **Ablation study** on γ and E for self-speculative decoding.

Table 7: Results on video understanding tasks with LLaVA-Video

Method	VideoMME				EgoSchema		LongVideo Bench	MVBench	Avg.	
	Short	Medium	Long	Overall	Subset	Total			Acc.	Rel. Acc. (%)
100% Tokens										
LLaVA-Video	77.0	62.1	53.3	64.2	59.4	57.3	59.5	61.9	60.7	100.0
Retain 25% Tokens										
FastV	71.7	59.2	50.9	60.6	56.0	54.8	56.4	59.1	57.7	95.1
VisionZip	74.0	60.3	52.9	62.4	59.0	57.0	58.3	60.0	59.4	97.9
FastVID	74.7	60.1	53.6	62.8	57.4	55.4	58.2	60.5	59.2	97.5
FlashVID	74.2	61.4	51.6	62.4	59.2	56.6	59.1	60.2	59.6	98.2
FlashPruner	75.6	61.8	53.0	63.5	60.0	57.2	60.2	61.4	60.6	99.8
Retain 20% Tokens										
FastV	69.3	58.3	49.9	59.2	54.8	54.1	56.0	58.4	56.9	93.7
VisionZip	72.3	59.6	53.3	61.7	59.0	56.4	58.0	59.8	59.0	97.2
FastVID	74.6	60.8	52.3	62.6	57.0	55.0	57.1	60.2	58.7	96.7
FlashVID	74.1	60.0	52.3	62.2	58.4	56.4	58.7	59.8	59.3	97.7
FlashPruner	74.8	61.1	52.9	62.9	58.8	57.0	59.4	61.2	60.1	99.0
Retain 15% Tokens										
FastV	67.9	56.9	50.8	58.5	52.8	53.1	54.5	57.5	55.9	92.1
VisionZip	72.9	58.1	51.9	61.0	58.6	55.7	57.2	59.6	58.4	96.2
FastVID	73.4	58.1	51.8	61.1	56.8	54.1	57.7	60.3	58.3	96.0
FlashVID	73.8	59.6	52.1	61.8	57.8	55.8	58.3	60.4	59.1	97.4
FlashPruner	74.6	60.9	52.8	62.8	58.6	56.5	58.9	61.0	59.8	98.5
Retain 10% Tokens										
FastV	64.3	53.8	49.2	55.8	50.6	51.1	53.6	56.2	54.2	89.3
VisionZip	69.4	57.9	51.2	59.5	54.4	53.9	54.5	58.5	56.6	93.2
FastVID	71.8	57.3	50.2	59.8	54.8	52.4	56.9	59.3	57.1	94.1
FlashVID	72.2	59.1	51.2	60.9	57.2	54.9	57.7	59.3	58.2	95.9
FlashPruner	73.2	60.7	52.0	62.0	58.0	56.1	58.6	60.4	59.3	97.7

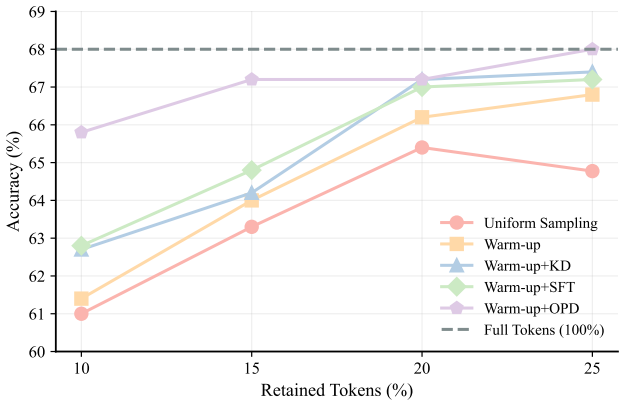


Figure 7: **Comparison on different training methods.** Uniform sampling denotes we directly uniform sample tokens at inference without any adaptation and training. We report results on MVBench with Qwen2.5-VL-7B.

Table 8: Latency statistics per sample on MVBench with Qwen2.5-VL-7B. TTFT denotes time-to-first-token and TPS denotes tokens per second during the decoding stage.

	ViT & projector ms	Token Pruning ms	LM Prefill ms	TTFT ms	# Vision Tokens –
Baseline	357.9	–	288.6	646.5	3815
FlashVID	357.9	48	54.7 (5.3x)	460.6 (1.4x)	382
FlashPruner	357.9	11	54.7 (5.3x)	423.6 (1.5x)	382

	TPS token/s	Decoding Time ms	Decoding Time w/CoT ms
Baseline	59.5	103	1859
FlashVID	60.0	102	1962
FlashPruner	116.7(1.96x)	52.5	1011

Table 9: Results with Qwen3-VL-8B.

	MMMU	**MMBench**	**RealWorldQA**	**MMstar**
Qwen3-VL-8B	69.6	85.0	71.5	70.9
FlashPruner (20% tokens)	71.2	83.4	70.8	70.4
FlashPruner (10% tokens)	70.6	81.7	69.2	69.8

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, claims made in the abstract and introduction accurately reflect the paper's contributions and scope

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Sec. H in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Yes, for all theoretical results, we provide the full set of assumptions and complete proofs in Sec. F in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details to reproduce our experimental results in Sec. 4 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will consider releasing code and model checkpoints after the submission is accepted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide all training and test details in Sec. 4 and Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Most of the experiments we report results with a single run due to the large computational resources required.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on the computer resources for efficiency analysis in the main text and for training in Tab. 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential social impacts in Sec. H in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We correctly cite all papers about data, code and models.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.