FlashVLM: Exploiting Layer Redundancy via Visual Attention for Efficient Vision-Language Inference

Sreetama Sarkar[1*], **Saket Gurukar**[3†], **Yuexi Zhang**[2], **Monami Banerjee**[2], **Ashwin Chandra**[2]

[1]University of Southern California [2]Samsung Research America [3]Microsoft
sreetama@usc.edu, {yuexi.zhang, m.banerjee, ashwin.c}@samsung.com

Abstract

Vision Language Models (VLMs) are rapidly advancing in multimodal understanding, largely due to increases in training data and model size. However, the computational demands of VLMs are also increasing, presenting challenges for their deployment in resource-constrained environments like smart televisions (TVs). VLMs are particularly useful in smart TVs for applications such as AI summarization, content question-and-answer (QnA), and user interface (UI) understanding and navigation. Previous research has focused on leveraging layer redundancy to accelerate VLM inference, but these methods often depend on extensive training, which requires significant computational resources and time. Recently, some inference-based solutions for layer skipping in Large Language Models (LLMs) have been proposed. In this paper, we demonstrate that the metrics used for layer skipping in LLMs do not always yield favorable results when applied to VLMs. We introduce an **inference-only layer skipping** strategy for VLMs based on attention to image tokens. Our approach increases **overall throughput by up to 21%** while maintaining close to baseline performance.

Introduction

Vision–Language Models (VLMs) have recently achieved remarkable performance across a wide range of multimodal understanding tasks, enabling applications such as ***AI-powered content summarization***, ***interactive UI navigation***, and ***voice-driven search***. These capabilities are increasingly being integrated into edge devices such as smart TVs, set-top boxes, and home assistants, bringing advanced AI functionality directly into consumer products. However, as VLM architectures advance, model sizes and computational demands continue to grow, making it infeasible to deploy state-of-the-art (SoTA) models on such devices. Edge environments are inherently resource-constrained in terms of processing power, memory, and energy budget, and thus cannot directly support the inference costs of large-scale VLMs without specialized efficiency strategies.

To facilitate the deployment of large-scale models on edge devices, researchers have explored a variety of efficiency techniques, including pruning (Ma, Fang, and Wang 2023; Sun et al. 2024), quantization (Frantar et al. 2022; Lin et al. 2024), efficient attention (Wang et al. 2020; Zhang et al. 2025a), and input sparsification methods. Token-dropping (Arif et al. 2025; Sarkar et al. 2025b), or selectively removing less informative input tokens, has been widely investigated to exploit the redundancy in images and videos. More recently, studies have revealed that layer redundancy also exists across transformer layers in these models (Elhoushi et al. 2024; Gromov et al. 2025), presenting an opportunity for substantial inference-time reduction. Existing layer skipping approaches are predominantly training-based, requiring expensive retraining or fine-tuning of the base model, which is impractical under limited GPU resources. Recently, inference-time solutions based on similarity metrics have shown promising results in Large Language Models (LLMs). While efficiency strategies from LLMs are often adapted to VLMs, we find that similarity metrics effective for LLMs do not yield favorable results when applied directly to VLMs, **underscoring the need for VLM-specific, training-free layer skipping methods.**

VLMs have been observed to sometimes overlook the visual context, producing incorrect outputs due to reduced attention to image tokens. Building on this insight, we propose a training-free layer skipping strategy that uses reduced visual attention to identify unimportant layers in the model. In transformer-based LLMs and VLMs, inference consists of two distinct stages: *prefill*, where the model processes the entire input sequence once to populate the key–value (KV) caches, and *decode*, where tokens are generated autoregressively, one at a time, using these cached states. While prefill cost scales with the prompt length, decode cost scales linearly with the number of generated tokens. As shown in Figure 1, for our target vision–language workloads, decode latency overtakes prefill latency once the number of generated tokens per image exceeds 10. Many downstream applications — such as image captioning, content summarization, and conversational assistants — naturally involve longer outputs, placing them firmly in this decode-dominated regime. We therefore focus on accelerating decode to improve end-to-end throughput. To this end, we introduce **FlashVLM, a visual attention–guided layer skipping strategy** that dynamically removes redundant computation during decoding, yielding substantial speedups without degrading output quality. Notably, our method is orthogonal to other efficiency techniques such as quantization and

[*]Work done during internship at Samsung Research America
[†]Work done while at Samsung Research America

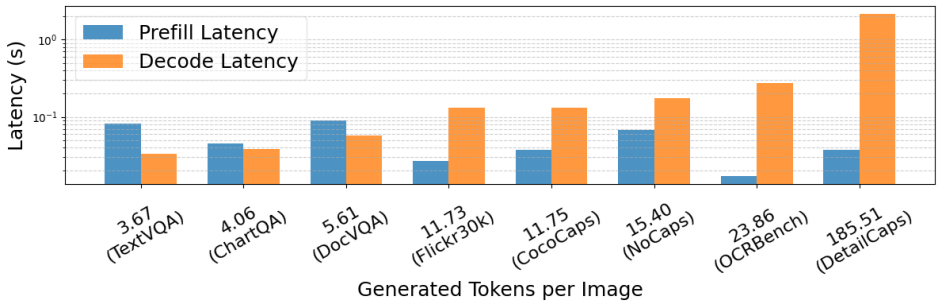


Figure 1: Prefill and decode latency per input sample averaged across 100 samples from different VLM benchmarks encompassing image captioning and visual QnA tasks. Latency values are measured on a single NVIDIA L40S GPU.

token dropping, and can be seamlessly combined with them for further performance improvements.

Our Contributions *Firstly*, we observe that **unlike LLMs, deeper layers in VLMs significantly impact model performance.** We conduct a detailed analysis of existing similarity metrics used for determining layer importance. We demonstrate that the metrics used for layer skipping in LLMs do not always yield favorable results when applied to VLMs. We introduce **FlashVLM, an inference-only layer skipping strategy for VLMs** based on attention to image tokens. *Secondly*, we perform a latency breakdown for the two stages of generation: prefill and decode. Our analysis reveals that the decode stage is the primary latency bottleneck for generation tasks, and we aim to perform layer skipping during decoding to maximize overall throughput. *Finally*, we demonstrate results across a wide range of visual question answering (QnA) and image captioning benchmarks, demonstrating the effectiveness of our proposed method. FlashVLM improves overall throughput by up to 21% with performance degradation of 1.7%.

Related Works

Recent advancements in optimizing large-scale models have focused on diverse strategies like pruning, quantization, linear attention, and activation sparsification. This section will specifically focus on methods that address layer redundancy, discussing both training-based and training-free solutions.

Training-based Methods: Training-based methods require training or fine-tuning the model to correctly interpret and make predictions with skipped computations. **Early-exit** frameworks allow for bypassing later layers in a model, based on the observation that models often reach the correct answer well before the final layer (Elhoushi et al. 2024). Early-exit methods have been extensively explored for language modeling for encoder-only transformers like BERT (Xin et al. 2020; Zhou et al.) and encoder-decoder architectures like T5 (Schuster et al. 2022; Elbayad et al. 2020). They rely either on confidence measures (Schuster et al. 2022) or train additional modules or classification heads (Elbayad et al. 2020; Schuster et al. 2022) to interpret the output of intermediate layers. Layerskip (Elhoushi et al. 2024) introduces an early-exit strategy for LLMs combined with speculative decoding to correct potential errors from exiting early. (Raposo et al. 2024) introduced **Mixture-of-depth** (MoD), which dynamically routes only a fraction of tokens through each layer instead of fully skipping layers. This facilitates that different tokens can require different levels of processing, reducing compute without affecting performance. For VLM models, layer skipping approaches can be applied either in the vision encoder or the language decoder. Videollm-MoD (Wu et al. 2024) applies MoD to the vision encoder, whereas Gamma-MoD (Luo et al. 2025) applied MoD to the language decoder. Unlike previous MoD methods, Gamma-MoD introduces a shared router across all layers for token selection and avoids skipping the question tokens. Notably, both early-exit frameworks and MoD methods involving fine-tuning routers and model parameters involve substantial GPU resources.

Training-free Methods: Token-dropping methods have been widely adopted to accelerate VLM inference by removing redundant tokens either at the output of the vision encoder (Arif et al. 2025) or a few layers into the LLM decoder (Chen et al. 2024; Zhang et al. 2025b). While inference-based strategies have been explored for addressing layer redundancy in language modeling, similar approaches for VLMs are still lacking. For instance, (Dalvi et al. 2020) analyzes the neuron-level redundancy in BERT (Devlin et al. 2019) models. ShortGPT (Men et al. 2025) identifies layers of interest in LLMs by measuring the cosine similarity between the input and the output of the layer, and skips layers during the prefilling stage for generative tasks. UIDL (Gromov et al. 2025) conducts a search to identify blocks of layers in LLMs that show the least deviation in angular distance between their input and output, ultimately eliminating the least important block. Notably, both these approaches demonstrate that deeper layers in LLMs are highly ineffective and redundant.

Our Approach

VLM Preliminaries

VLMs typically comprise four key components: a text tokenizer, a vision encoder, a projector, and a language decoder. The text tokenizer processes the language input by segmenting it into discrete tokens and converting them into text embeddings, referred to as *text tokens*. Similarly, the vision encoder partitions the input image into patches and transforms them into corresponding visual embeddings. These visual embeddings are then mapped into the text embedding space

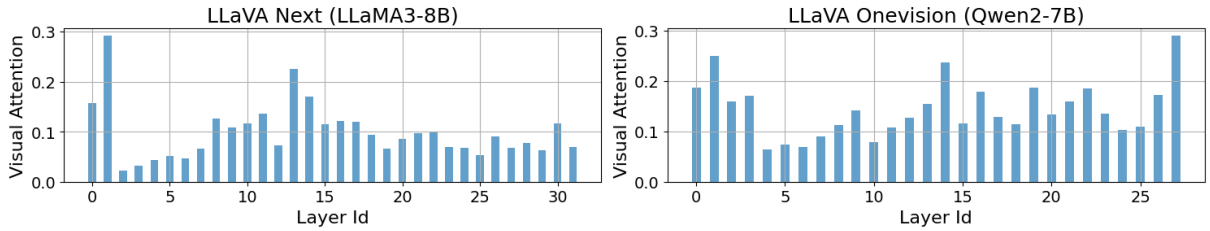


Figure 2: Layerwise visual attention averaged across all query tokens for 500 sample images from OCRBench.

through the projector, producing *vision tokens*. Finally, the vision and text tokens are concatenated and fed into the language decoder. The majority of the computation in a VLM is concentrated in the vision encoder and the language decoder. However, the vision encoder processes each image only once, whereas the language decoder iteratively processes tokens for every generated output token. Therefore, our efficiency strategy focuses on skipping layers in the language decoder.

The language decoder consists of a stack of transformer layers, each consisting of a multi-head attention (MHA) block followed by a feed-forward network (FFN). The MHA takes as input N tokens, consisting of both text and vision tokens, each represented as an embedding of dimension d, $X \in \mathbb{R}^{N\times d}$. These are mapped into Query, Key and Value matrices $(Q, K, V) \in \mathbb{R}^{N\times d}$. MHA then computes low-dimensional projections of (Q, K, V) given by $(Q^i, K^i, V^i) \in \mathbb{R}^{N\times d_k}$ for each head i where $d_k = d/H$, H denoting the number of attention heads. The scaled dot product attention for each head is then computed using

$$A^i = \text{Softmax}\left(\frac{Q^i K^{iT}}{\sqrt{d_k}}\right) \tag{1}$$

Here, $A \in \mathbb{R}^{N\times N}$ denotes the attention of query tokens Q^i to key tokens for the i^{th} attention head.

Layer Skipping using Visual Attention

Visual attention denotes the fraction of the total attention that the model pays to the input context provided by the image to generate the next token. Previous research (Liu, Zheng, and Chen 2024; Sarkar et al. 2025a) has shown that low visual attention causes the model to ignore the input context and generate incorrect or hallucinated outputs. Based on this insight, ***we hypothesize that layers with low visual attention do not significantly contribute to generating the correct output, and can be skipped to reduce computations without compromising performance***.

We define ***visual attention*** as the attention between the current text query token $q^i \in \mathbb{R}^{1\times d_k}$ and the vision tokens, given by Equation 2. I_{start} and I_{end} denote the start and end indices of vision tokens. The visual attention is averaged across all heads. We define ***layer importance score*** as the sum of attention scores for that layer across all vision tokens.

$$\text{A}_v = \sum_{j=I_{\text{start}}}^{I_{\text{end}}} \left(\frac{1}{H}\sum_{i=1}^{H} \text{A}^i[-1, j]\right) \tag{2}$$

We demonstrate the visual attention scores across different layers for the LLaVA models in Figure 2. We observe that despite vision tokens constituting 99% of the input, they receive <10% of the overall attention for most layers in LLaMA, and <20% in Qwen, with some layers exhibiting particularly low visual attention scores. We propose to identify and skip layers with the lowest A_v scores. The detailed algorithm for our layer skipping is given in Algorithm 1.

Algorithm 1: Layer Selection based on Visual Attention

Input: Sample set $\mathcal{S}$, total number of decoder layers L, skip ratio $r \in [0, 1]$
Output: Selected layer indices to skip

Initialize list of layerwise attention sums: $L_{attn\_sum} \leftarrow [0] \times L$
Initialize total number of tokens: $n_{tokens} \leftarrow 0$
for each sample $s \in \mathcal{S}$ **do**
  for each generated token t for s **do**
    $n_{tokens} \leftarrow n_{tokens} + 1$
    for each layer l in range(L) **do**
      $A_{v,l}$ = attention score of current query token q_t to image tokens for layer l based on Equation 2
      $L_{attn\_sum}[l] \leftarrow L_{attn\_sum}[l] + A_{v,l}$
    end for
  end for
end for
for each layer l in range(L) **do**
  $L_{attn\_avg}[l] \leftarrow L_{attn\_sum}[l]/n_{tokens}$
end for
Sort L_{attn_avg} in increasing order
$k \leftarrow \lfloor r \times L \rfloor$
return first k layer indices from sorted L_{attn_avg}

Comparing with other Layer Skipping Metrics

We compare layer importance based on visual attention with layer importance metrics proposed in literature based on magnitude, direction of vectors and depth-wise approaches.

L2 Norm: The L2 norm difference gives a notion of the difference in magnitude between a layer's input and output. This difference between the layer's input X_{in} and output X_{out} can be calculated using Equation 3.

$$\text{norm\_diff}(X_{in}, X_{out}) = abs(\|X_{\text{in}}\|_2 - \|X_{\text{out}}\|_2) \tag{3}$$

Cosine Similarity: LLM models utilize pre-normalization before processing each layer. Prior research (Men et al. 2025) suggests that the magnitude of the output becomes less significant in this context. As a result, they advocate for the use of magnitude-agnostic metrics, such as cosine similarity (Men et al. 2025) or angular distance (Gromov et al.

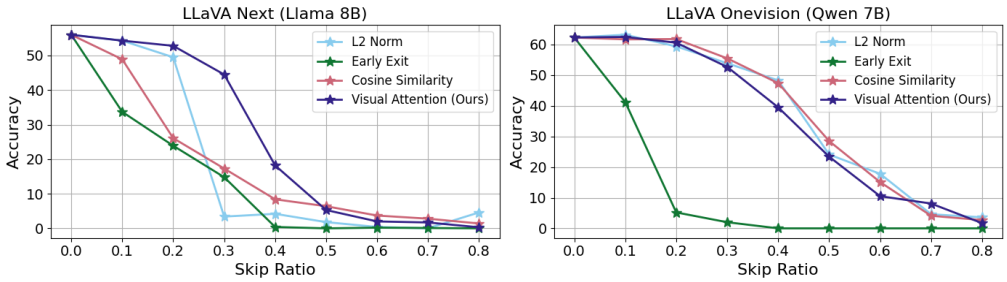


Figure 3: Accuracy on OCRBench for different layer skip ratios using different skip metrics.

2025), which focus on measuring the direction between vectors. The cosine similarity or angular distance between a layer's input and output can be calculated using Equations 4 and 5.

$$\text{cosine\_similarity}(X_{\text{in}}, X_{\text{out}}) = \frac{X_{\text{in}} \cdot X_{\text{out}}}{\|X_{\text{in}}\|\|X_{\text{out}}\|} \quad (4)$$

$$\text{angular\_dist}(X_{\text{in}}, X_{\text{out}}) = \frac{1}{\pi} arccos\left(\frac{X_{\text{in}} \cdot X_{\text{out}}}{\|X_{\text{in}}\|\|X_{\text{out}}\|}\right) \quad (5)$$

Early-exit: In early-exit frameworks, layer importance is estimated based on depth, with deeper layers being less important. Previous research on layer skipping in LLMs has found that deeper layers are often insignificant or redundant (Elhoushi et al. 2024; Gromov et al. 2025).

Experimental Results

Experimental Setup

Models and Datasets: We demonstrate results for LLaVA Onevision (Li et al. 2024) and LLaVA-Next (Liu et al. 2024a) with two different LLMs LLaMA-8B (Touvron et al. 2023) and Qwen-7B (Bai et al. 2023). We use 500 samples from OCRBench as our calibration set to determine layer importance based on visual attention as described in the previous section. We demonstrate results across 7 benchmarks, including image captioning tasks like NoCaps (Agrawal et al. 2019), CocoCaps (Chen et al. 2015), Flickr30k (Young et al. 2014), and DetailCaps (Dong et al. 2024), and visual QnA benchmarks like OCRBench (Liu et al. 2024b), ChartQA (Masry et al. 2022) and TextVQA (Singh et al. 2019). For throughput estimation, we use 100 samples from each benchmark. Evaluation is performed on NVIDIA L40S GPUs.

Evaluation Metrics: For throughput measurement, we report the following metrics:

- *Prefill time/sample*: The average time required to process the entire input sequence and generate the first output token for a single input sample.
- *Decode time/sample*: The average time required to generate all subsequent output tokens (after the first token) for a single input sample.
- *Decode time/token*: The total decoding time divided by the total number of output tokens generated (excluding the first token).
- *Overall throughput (tokens/s)*: The total number of output tokens generated (including the first token) divided by the total time for both prefill and decode stages.

Baselines: We compare FlashVLM with two SoTA inference-only layer skipping methods, ShortGPT (Men et al. 2025) and UIDL (Gromov et al. 2025). ShortGPT uses *cosine similarity* as described in Equation 4 to determine layer importance, and proposes skipping only during *prefill* stage for generative tasks. UIDL skips a block of consecutive layers instead of individual layers. It conducts a search with a block size that matches the skip ratio, identifying the block with the least deviation in angular distance (as described in Equation 5) between its input and output.

Evaluating different layer skipping metrics

We compare different layer skipping metrics in Figure 3. We skip layers during the decode phase and plot the accuracy vs skip ratio on OCRBench for LLaMA and Qwen backbones. We observe that early-exit or skipping deeper layers leads to a sharp drop in accuracy even when only 10% of the layers are skipped, equating to 2 layers for Qwen and 3 layers for LLaMA. This is a key difference with LLM models where deeper layers have been found to highly ineffective (Gromov et al. 2025). Interestingly, using cosine similarity or L2 norm difference to determine layer importance yields promising performance for Qwen, but leads to a decline in accuracy for LLaMA, suggesting these are not consistent metrics across LLMs. *We find that cosine similarity identifies deeper layers for LLaMA, whereas L2 norm difference skips the first layer, which likely contributes to the observed accuracy drop.* On the other hand, our proposed visual attention approach shows promising results for both models, with no significant decrease in accuracy when skipping up to 20% of the layers. We will demonstrate results from several other benchmarks in the following sections.

Results and Analysis

Evaluation on Image Captioning: We present results for image captioning tasks in Table 1. For our proposed strategy, prefill latency remains unchanged since we do not skip layers during the prefill stage. We obtain latency improvements in decoding proportional to the number of skipped layers. For example, for a skip ratio of $\sim$20% for LLaMA decoder, we observe a reduction in decode time/token by 15% for Flickr30k to 18% for NoCaps and CocoCaps. Similarly, for Qwen-7B backbone, we obtain a decoding latency reduction, ranging from from 17% to 21%.

Overall throughput is strongly affected by the length of the generated output. In our measurements, prefill latency is typically 4 to 5 times higher than the per-token decode

Model	Dataset	Method	Skip Ratio	Rouge_L (↑)	Throughput Estimation			
					Prefill time/ sample (ms) (↓)	Decode time/ token (ms) (↓)	Avg length	Throughput (tokens/s)(↑)
LLaVA-Next (LLaMA3-8B)	NoCaps	Baseline	0.0	0.6146	50.7	12.9	12.5	22.09
		ShortGPT	0.20	0.0503	14.8	13.0	64	34.17
		UIDL	0.20	0.2527	**14.8**	10.8	29.5	34.88
		Ours	0.20	**0.6005**	49.9	**10.6 (-17.8%)**	13.6	**25.58 (+15.8%)**
	Flickr30k	Baseline	0.0	0.5317	40.3	12.8	12.8	24.55
		ShortGPT	0.20	0.0354	**13.4**	12.7	64	35.12
		UIDL	0.20	0.2446	13.6	**10.8**	23.3	34.78
		Ours	0.20	**0.5301**	40.7	10.9 (-14.8%)	13.4	**27.71 (+12.9%)**
	CocoCaps	Baseline	0.0	0.6067	48.9	12.9	12.4	22.63
		ShortGPT	0.20	0.0365	15.0	13.1	64	34.32
		UIDL	0.20	0.2282	**14.7**	10.7	27.6	34.77
		Ours	0.20	**0.5898**	47.3	**10.6 (-17.8%)**	12.8	**25.78 (+13.9%)**
LLaVA Onevision (Qwen2-7B)	NoCaps	Baseline	0.0	0.5526	68.3	12.0	15.4	15.276
		ShortGPT	0.20	0.1742	11.7	12.0	5.21	8.05
		UIDL	0.20	**0.5958**	**11.7**	10.0	11.2	14.86
		Ours	0.20	0.528	67.7	**9.8 (-18.3%)**	16.7	**17.47 (+14.4%)**
	Flickr30k	Baseline	0.0	0.5363	26.9	12.0	11.7	23.98
		ShortGPT	0.20	0.1542	11.6	12.0	7.98	21.82
		UIDL	0.20	0.5361	**11.7**	**10.0**	11.3	**27.65**
		Ours	0.20	**0.5375**	26.9	**10.0 (-16.7%)**	11.9	26.51 (+10.6%)
	CocoCaps	Baseline	0.0	0.611	37.2	12.3	11.8	20.06
		ShortGPT	0.20	0.1491	11.9	10.0	8.7	21.97
		UIDL	0.20	0.5754	**11.7**	9.9	11	**22.79**
		Ours	0.20	**0.6024**	36.7	**9.7 (-21.14%)**	11.9	22.11 (+10.22%)
	DetailCaps	Baseline	0.0	0.3024	37.6	11.6	185.5	38.34
		ShortGPT	0.20	0.0467	11.4	11.5	43.6	33.08
		UIDL	0.20	**0.2885**	**11.4**	**9.6**	149.5	**44.83**
		Ours	0.20	0.2864	38.0	**9.6 (-20%)**	147.7	44.32 (+15.6%)

Table 1: Image Caption Evaluation on LLaVA with two different backbones. The best values for each column are highlighted in bold. If the average length of the generated output tokens is approximately double or more than the average length of the ground truth captions, it is marked in red, indicating the presence of repetitions or nonsensical tokens.

latency, though the exact ratio depends on the input image size. When the generation is long, decode latency dominates the total time, making it the primary factor for improving tokens-per-second. In contrast, for shorter outputs with only a few tokens, prefill latency forms a large share of the total latency, which limits overall throughput.

We also find that skipping layers can unintentionally increase generation length by producing repetitive or unnecessary tokens. This artificially inflates throughput for that task, but at the cost of correctness. Such behavior is particularly pronounced in LLaMA when using methods like ShortGPT and UIDL, where the higher throughput numbers do not reflect usable outputs. Hence, we mark these numbers with red in Tables 1 and 2. We suspect that this occurs since deeper layers in LLaMA are deemed less critical by these approaches, based on cosine similarity and angular distance metrics, and skipped, leading to degraded output quality.

Our approach achieves an overall throughput improvement of up to 15.8% for LLaMA-3 and 15.6% for Qwen-2 backbones. UIDL achieves the best throughput performance trade-off for captioning tasks for Qwen. However, it suffers a significant accuracy drop for LLaMA. ShortGPT is observed to perform poorly for both models. Even though it skips layers only during prefill and not during decoding, this appears to negatively impact performance, as those layers cannot leverage the image context since the necessary input token representations were never cached.

Evaluation on Visual QnA: We present evaluation results for visual QnA tasks in Table 2. We observe that both ShortGPT and UIDL suffer a substantial drop in accuracy for both model backbones as the task complexity increases, with ShortGPT attaining close to zero accuracy in many cases. On the contrary, our approach shows promising results on these tasks with performance drops of 1.7%, 0.44% and 1.01% for OCRBench, ChartQA and TextVQA on Qwen. For LLaMA, FlashVLM degrades performance by 3.2%, 4.3% and 1.4% for OCRBench, ChartQA and TextVQA respectively. We obtain throughput improvements of up to 20.8% for OCRBench with a performance degradation of 1.7%. The throughput gains for TextVQA and ChartQA are limited due to their very short generation lengths. However, most real-world tasks like content summarization and UI

Model	Dataset	Method	Skip Ratio	Accuracy (↑)	Throughput Estimation			
					Prefill time/ sample (ms) (↓)	Decode time/ token (ms) (↓)	Avg length	Throughput (tokens/s)(↑)
LLaVA-Next (LLaMA3-8B)	OCRBench	Baseline	0.0	0.56	21.4	12.8	33.7	35.89
		ShortGPT	0.20	0.002	12.2	13.2	128	38.85
		UIDL	0.20	0.073	**12.0**	**10.6**	95.3	45.88
		Ours	0.20	**0.528**	21.4	11.0 (-14.1%)	16.9	**38.01 (+5.9%)**
	ChartQA	Baseline	0.0	0.6872	42.1	13.4	4.1	14.11
		ShortGPT	0.20	0.0	14.4	13.3	16	27.91
		UIDL	0.20	0.1268	**14.2**	**10.7**	12.4	28.99
		Ours	0.20	**0.644**	41.8	10.8 (-19.4%)	4.4	**15.29 (+8.4%)**
	TextVQA	Baseline	0.0	0.6202	49.6	13.2	5.0	14.08
		ShortGPT	0.20	0.0	15.1	13.2	1024	36.97
		UIDL	0.20	0.1292	**14.9**	**10.8**	245.2	42.71
		Ours	0.20	**0.6064**	50.2	10.9 (-17.4%)	4.8	**14.26 (+1.3%)**
LLaVA Onevision (Qwen2-7B)	OCRBench	Baseline	0.0	0.623	17.0	12.0	23.8	35.42
		ShortGPT	0.20	0.049	**11.5**	12.0	46.4	22.45
		UIDL	0.20	0.445	11.7	10.0	47.4	**45.86**
		Ours	0.20	**0.606**	17.0	**9.9 (-17.5%)**	32.0	42.8 (+20.8%)
	ChartQA	Baseline	0.0	0.8004	45.0	12.3	4.1	8.96
		ShortGPT	0.20	0.0084	**12.0**	11.9	9.1	17.25
		UIDL	0.20	0.3868	**12.0**	10.0	3.5	9.03
		Ours	0.20	**0.796**	44.0	**10.0 (-18.67%)**	4.0	**9.24 (+3.13%)**
	TextVQA	Baseline	0.0	0.7593	81.0	12.5	3.7	5.32
		ShortGPT	0.20	0.0005	**20.0**	11.6	59.1	27.78
		UIDL	0.20	0.59	**20.0**	10.0	3.9	**6.36**
		Ours	0.20	**0.7492**	79.0	**10.0 (-20%)**	3.6	5.42 (+1.9%)

Table 2: Visual QnA evaluation on LLaVA with two different backbones. The best values are bolded. The average length of the generated output, substantially exceeding the ground truth, is marked in red, indicating repetitive or nonsensical tokens.

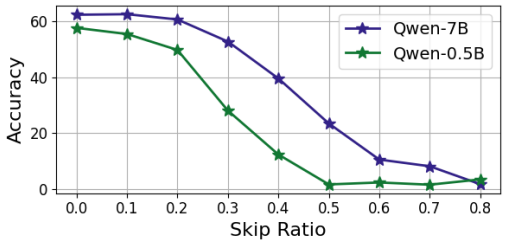


Figure 4: Accuracy vs skip ratio on OCRBench for LLaVA Onevision with Qwen-7B and Qwen-0.5B.

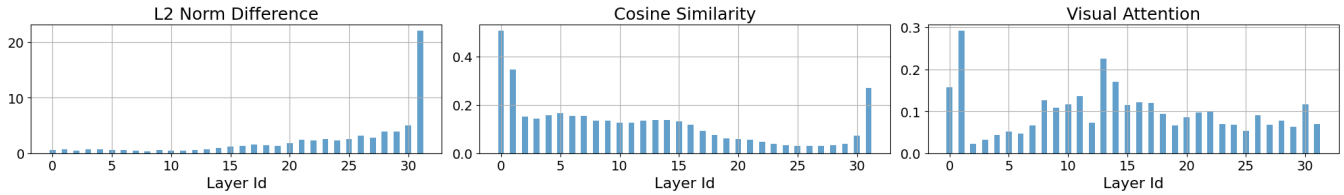
navigation require substantially longer output, in which case our approach offers greater benefits.

Layer skipping for different model sizes: We present accuracy vs skip ratios using FlashVLM for LLaVA Onevision with Qwen-7B and Qwen-0.5B in Figure 4. This shows that bigger models are more resilient to layer skipping, suggesting higher layer redundancy in bigger models.

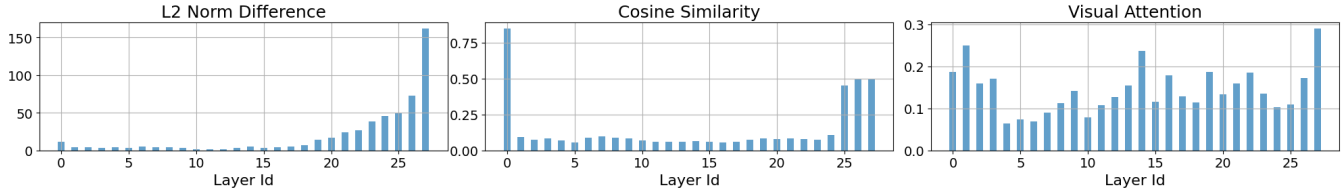
Layer Importance Score Visualization: We visualize the layerwise scores based on different metrics like L2 norm difference, cosine similarity and visual attention for LLaVA-Next (LLaMA3-8B) and LLaVA Onevision (Qwen2-7B) based on 500 samples from OCRBench (Liu et al. 2024b) in Figure 5a and 5b respectively. We observe that later layers exhibit high values for *L2 norm difference* for both models, with the middle layers having the lowest scores. However, for LLaMA, the first layer also has one of the lowest values of L2 norm difference, which leads to the first layer being skipped early on when L2 norm difference is the chosen metric. *Cosine similarity* is high for the initial and last three layers for Qwen, with middle layers having low values. However, for LLaMA, it shows the lowest values for the later layers except the last layer. This probably leads to the observed accuracy drop on LLaMA using cosine similarity, since we find that layer layers play a key role in maintaining VLM performance. For *visual attention*, lowest scores are observed for layer ids 2 to 6 for LLaMA and for layer ids 4 to 7 and 10 for Qwen.

To further verify if the attention patterns are consistent across benchmarks, we plot the layerwise visual attention scores for LLaMA3-8B on different benchmarks in Figure 5c. We observe that for the same model, similar attention patterns are observed across benchmarks. This suggests that our attention-based layer identification strategy, based on a subset of data samples for a particular task, can generalize effectively across different benchmarks.

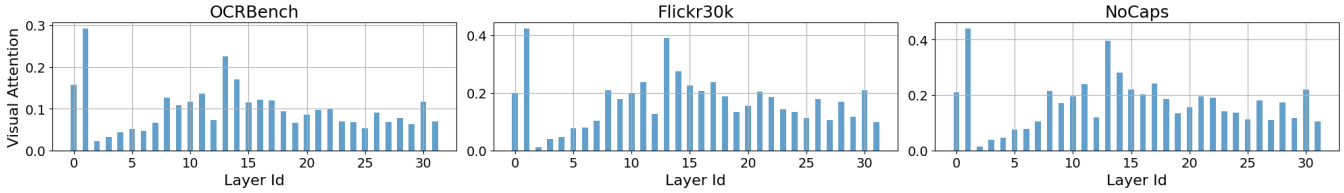
Comparison with token merging methods: For token merging approaches, we compare our method against ToMe (Token Merging) (Bolya et al. 2022) and PiToMe (Protect Informative Tokens before Merging) (Tran et al. 2024), which exploit token redundancy to increase through-



(a) Layer scores for LLaVA-Next (LLaMA3-8B) based on L2 norm difference (left), cosine similarity (middle), and visual attention (right)



(b) Layer scores for LLaVA Onevision (Qwen2-7B) based on L2 norm difference (left), cosine similarity (middle), and visual attention (right)



(c) Layer scores for LLaVA-Next (LLaMA3-8B) based on visual attention for different benchmarks: OCRBench (left), Flickr30k (middle), and NoCaps (right). We observe similar trends in layer scores across all benchmarks.

Figure 5: Layer score visualization for different layer skipping metrics on two different models, across three benchmarks.

put without requiring additional training. ToMe progressively merges similar tokens within a transformer using a lightweight matching algorithm, whereas PiToMe extends ToMe by prioritizing the preservation of informative tokens through an additional graph-energy–based metric. r controls the number of tokens merged per layer in ToMe, while in PiToMe, r denotes the fraction of tokens preserved in each layer. Notably, both ToMe and PiToMe were originally developed for Vision Transformers (ViTs). We extend these techniques to reduce the number of vision tokens in VLMs, and present the results in Table 3. Generally, we drop tokens both in the base image and sub-images, whereas, for "w/o base" we drop tokens only in the sub-images. While these methods yield throughput improvements, they suffer significant accuracy degradation, making them unusable for complex tasks like OCRBench and ChartQA.

Dataset	Method	Accuracy	Throughput Gain
OCR-Bench	Baseline	62.3	1×
	ToMe (r=10)	36.4	1.11×
	ToMe w/o base (r=10)	57.5	1.08×
	PiToMe (r=0.95)	32.7	1.6×
	PiToMe w/o base (r=0.95)	55.4	1.4×
	Ours	**60.6**	1.21×
ChartQA	Baseline	80.04	1×
	ToMe (r=10)	40.2	1.3×
	ToMe w/o base (r=10)	64.6	1.5×
	PiToMe (r=0.95)	35.36	2.2×
	PiToMe w/o base (r=0.95)	62.28	1.8×
	Ours	**79.6**	1.03×

Table 3: Comparison of our approach with token merging methods on LLaVA Onevision (Qwen2-7B)

Conclusions and Future Work

This paper addresses the critical need for inference efficiency in VLMs by proposing an innovative, inference-only layer skipping strategy. Previous research has demonstrated that large foundation models exhibit significant layer redundancy, which can be leveraged to speed up inference. Notably, deeper layers have been found to be ineffective for LLMs. Our findings reveal that, unlike LLMs, deeper layers play a crucial role in the VLMs. Moreover, traditional layer skipping strategies are usually training-based, requiring substantial GPU resources. To address this, we propose an inference-only approach based on visual attention for layer skipping in VLMs. The proposed strategy exhibits promising results, achieving notable inference speedups while maintaining high performance across a diverse range of tasks. In future, we aim to extend this attention-based approach to the prefilling stage of VLM inference, which holds the potential for even greater throughput gains. Additionally, future work will focus on investigating why deeper VLM layers are more critical than their LLM counterparts.

References

Agrawal, H.; Desai, K.; Wang, Y.; Chen, X.; Jain, R.; Johnson, M.; Batra, D.; Parikh, D.; Lee, S.; and Anderson, P. 2019. nocaps: novel object captioning at scale. In *Proceedings of the IEEE International Conference on Computer Vision*, 8948–8957.

Arif, K. H. I.; Yoon, J.; Nikolopoulos, D. S.; Vandierendonck, H.; John, D.; and Ji, B. 2025. HiRED: Attention-Guided Token Dropping for Efficient Inference of High-Resolution Vision-Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 1773–1781.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. *arXiv preprint arXiv:2308.12966*.

Bolya, D.; Fu, C.-Y.; Dai, X.; Zhang, P.; Feichtenhofer, C.; and Hoffman, J. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Chen, L.; Zhao, H.; Liu, T.; Bai, S.; Lin, J.; Zhou, C.; and Chang, B. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, 19–35. Springer.

Chen, X.; Fang, H.; Lin, T.-Y.; Vedantam, R.; Gupta, S.; Dollar, P.; and Zitnick, C. L. 2015. Microsoft COCO Captions: Data Collection and Evaluation Server. arXiv:1504.00325.

Dalvi, F.; Sajjad, H.; Durrani, N.; and Belinkov, Y. 2020. Analyzing Redundancy in Pretrained Transformer Models. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Dong, H.; Li, J.; Wu, B.; Wang, J.; Zhang, Y.; and Guo, H. 2024. Benchmarking and Improving Detail Image Caption. *arXiv preprint arXiv:2405.19092*.

Elbayad, M.; Gu, J.; Grave, E.; and Auli, M. 2020. Depth-Adaptive Transformer. In *International Conference on Learning Representations*.

Elhoushi, M.; Shrivastava, A.; Liskovich, D.; Hosmer, B.; Wasti, B.; Lai, L.; Mahmoud, A.; Acun, B.; Agarwal, S.; Roman, A.; Aly, A.; Chen, B.; and Wu, C.-J. 2024. LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12622–12642. Bangkok, Thailand: Association for Computational Linguistics.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. GPTQ: Accurate Post-training Compression for Generative Pretrained Transformers. *arXiv preprint arXiv:2210.17323*.

Gromov, A.; Tirumala, K.; Shapourian, H.; Glorioso, P.; and Roberts, D. 2025. The Unreasonable Ineffectiveness of the Deeper Layers. In *The Thirteenth International Conference on Learning Representations*.

Li, B.; Zhang, Y.; Guo, D.; Zhang, R.; Li, F.; Zhang, H.; Zhang, K.; Li, Y.; Liu, Z.; and Li, C. 2024. LLaVA-OneVision: Easy Visual Task Transfer. *arXiv preprint arXiv:2408.03326*.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *MLSys*.

Liu, H.; Li, C.; Li, Y.; Li, B.; Zhang, Y.; Shen, S.; and Lee, Y. J. 2024a. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge.

Liu, S.; Zheng, K.; and Chen, W. 2024. Paying more attention to image: A training-free method for alleviating hallucination in lvlms. In *European Conference on Computer Vision*, 125–140. Springer.

Liu, Y.; Li, Z.; Huang, M.; Yang, B.; Yu, W.; Li, C.; Yin, X.-C.; Liu, C.-L.; Jin, L.; and Bai, X. 2024b. OCRBench: on the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12).

Luo, Y.; Luo, G.; Ji, J.; Zhou, Y.; Sun, X.; Shen, Z.; and Ji, R. 2025. $\gamma-$MoD: Exploring Mixture-of-Depth Adaptation for Multimodal Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*.

Masry, A.; Long, D.; Tan, J. Q.; Joty, S.; and Hoque, E. 2022. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2263–2279. Dublin, Ireland: Association for Computational Linguistics.

Men, X.; Xu, M.; Zhang, Q.; Yuan, Q.; Wang, B.; Lin, H.; Lu, Y.; Han, X.; and Chen, W. 2025. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *Findings of the Association for Computational Linguistics: ACL 2025*, 20192–20204. Vienna, Austria: Association for Computational Linguistics. ISBN 979-8-89176-256-5.

Raposo, D.; Ritter, S.; Richards, B.; Lillicrap, T.; Humphreys, P. C.; and Santoro, A. 2024. Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. arXiv:2404.02258.

Sarkar, S.; Che, Y.; Gavin, A.; Beerel, P. A.; and Kundu, S. 2025a. Mitigating Hallucinations in Vision-Language Models through Image-Guided Head Suppression. *arXiv preprint arXiv:2505.16411*.

Sarkar, S.; Datta, G.; Kundu, S.; Zheng, K.; Bhattacharyya, C.; and Beerel, P. A. 2025b. MaskVD: Region masking for efficient video object detection. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 1955–1964. IEEE.

Schuster, T.; Fisch, A.; Gupta, J.; Dehghani, M.; Bahri, D.; Tran, V. Q.; Tay, Y.; and Metzler, D. 2022. Confident Adaptive Language Modeling. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Singh, A.; Natarjan, V.; Shah, M.; Jiang, Y.; Chen, X.; Batra, D.; Parikh, D.; and Rohrbach, M. 2019. Towards VQA Models That Can Read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8317–8326.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2024. A Simple and Effective Pruning Approach for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tran, C.; MH Nguyen, D.; Nguyen, M.-D.; Nguyen, T.; Le, N.; Xie, P.; Sonntag, D.; Zou, J. Y.; Nguyen, B.; and Niepert, M. 2024. Accelerating transformers with spectrum-preserving token merging. *Advances in Neural Information Processing Systems*, 37: 30772–30810.

Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-Attention with Linear Complexity. arXiv:2006.04768.

Wu, S.; Chen, J.; Lin, K. Q.; Wang, Q.; Gao, Y.; Xu, Q.; Xu, T.; Hu, Y.; Chen, E.; and Shou, M. Z. 2024. VideoLLM-MoD: Efficient Video-Language Streaming with Mixture-of-Depths Vision Computation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xin, J.; Tang, R.; Lee, J.; Yu, Y.; and Lin, J. 2020. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2246–2251. Online: Association for Computational Linguistics.

Young, P.; Lai, A.; Hodosh, M.; and Hockenmaier, J. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2: 67–78.

Zhang, M.; Arora, S.; Chalamala, R.; Spector, B. F.; Wu, A.; Ramesh, K.; Singhal, A.; and Re, C. 2025a. LoLCATs: On Low-Rank Linearizing of Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

Zhang, Y.; Fan, C.-K.; Ma, J.; Zheng, W.; Huang, T.; Cheng, K.; Gudovskiy, D.; Okuno, T.; Nakata, Y.; Keutzer, K.; et al. 2025b. SparseVLM: Visual Token Sparsification for Efficient Vision-Language Model Inference. In *International Conference on Machine Learning*.

Zhou, W.; Xu, C.; Ge, T.; McAuley, J.; Xu, K.; and Wei, F. ???? BERT Loses Patience: Fast and Robust Inference with Early Exit. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, 18330–18341. Curran Associates, Inc.