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

Anonymous ACL submission

Abstract

Despite the advancements in Vision Language Models (VLMs) in multimodal understanding, the increasing computational demands pose challenges for deployment in resource-constrained platforms, such as smart TVs, for applications like content summarization and UI navigation. While prior work accelerates inference by exploiting layer redundancy, these methods often require costly retraining. Recent inference-based layer skipping techniques for Large Language Models (LLMs) do not directly translate to VLMs due to differing visual attention dynamics. To bridge this gap, we propose an attention-guided, inference-only layer skipping strategy for VLMs that leverages image token attention to determine layer importance. To address diverse tasks and deployment needs, our framework introduces two distinct modes: a ***static skipping*** strategy that utilizes offline calibration to identify universally redundant layers and incurs no runtime overhead, and a ***dynamic skipping*** strategy that evaluates per-sample layer importance during the prefill stage, enabling calibration-free adaptation to out-of-distribution inputs. Extensive experiments across multiple benchmarks show that our method improves overall throughput by up to 1.3× while maintaining performance close to the baseline.

1 Introduction

Vision–Language Models (VLMs) have 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 et al., 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

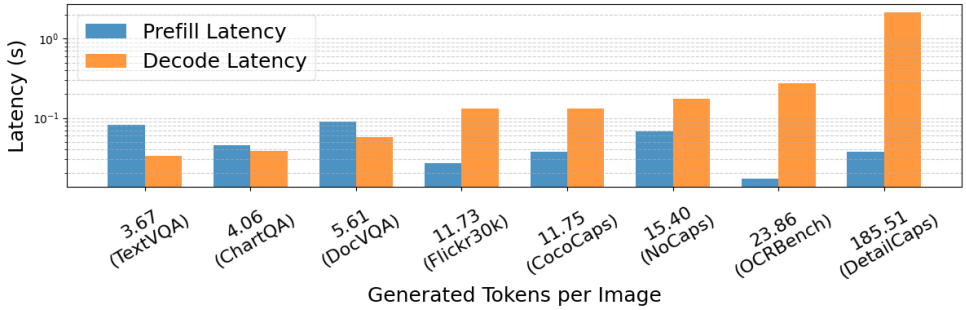


Figure 1: Prefill and decode latency per input sample averaged across 100 samples from different VLM benchmarks encompassing image captioning and visual QnA tasks. Latencies are measured on a single NVIDIA L40S GPU.

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, conversational assistants, and chain-of-thought (CoT) reasoning, 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 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 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. FlashVLM supports two distinct modes: *static skipping* that uses offline calibration to identify universally redundant layers with no runtime overhead, and *dynamic skipping* that estimates per-sample layer importance during inference for improved adaptability to diverse inputs. *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 1.3× while maintaining close to baseline performance.

2 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 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.

3 Our Approach

3.1 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 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 ith attention head.

3.2 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 et al., 2024b; 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***. To thoroughly explore this, we propose an attention-guided layer-skipping framework that can be deployed in two distinct modes: a highly efficient ***static skipping*** and an instance-aware ***dynamic skipping***.

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, averaged across all heads, where I_{start} and I_{end} denote the start and end indices of vision tokens. We define ***layer importance score***,

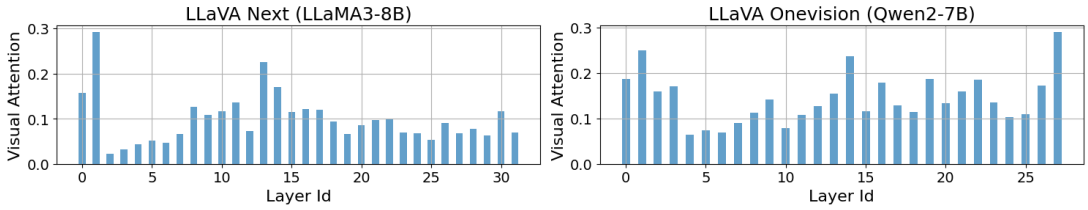


Figure 2: Layerwise visual attention averaged across all query tokens for 500 sample images from OCRBench.

given by Equation 2, as the sum of attention scores for that layer across all vision tokens.

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

We demonstrate the visual attention scores across different layers of 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. To exploit this, we propose skipping layers with the lowest A_v scores using one of the following two strategies:

Static (Calibration-Based) Skipping: In this offline mode, we calculate the average A_v scores across a small, representative calibration dataset to identify universally redundant layers in advance. During inference, these predefined layers are consistently skipped for all inputs. This approach is highly efficient, offering significant computational speedups with no run-time overhead for layer selection. The detailed algorithm for our layer skipping is given in the Algorithm 1.

Dynamic (Instance-Aware) Skipping: In this mode, we dynamically compute the A_v scores during the prefill stage for every sample. The layers exhibiting the lowest attention for that specific input are then skipped during the subsequent decoding phase. This strategy eliminates the need for prior calibration data and allows the model to adapt flexibly to out-of-distribution instances.

3.3 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 measures the change in magnitude between a layer's input X_{in} and output X_{out}, as computed in 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 each layer, making output magnitude less informative. Prior research (Men et al., 2025) recommends magnitude-agnostic metrics, such as cosine similarity (Men et al., 2025) or angular distance (Gromov et al., 2025), which measures the direction between a layer's input and output, as computed in Equations 4 and 5.

$$\text{cosine_similarity}(X_{\text{in}}, X_{\text{out}}) = \frac{X_{\text{in}} \cdot X_{\text{out}}}{\|X_{\text{in}}\| \|X_{\text{out}}\|} \tag{4}$$

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

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 redundant (Elhoushi et al., 2024; Gromov et al., 2025).

4 Experimental Results

4.1 Experimental Setup

Models and Datasets: We demonstrate results for Qwen3-VL (Bai et al., 2025), 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). For static skipping, 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., 2024c), and ChartQA (Masry et al., 2022). For throughput estimation, we use 100 samples from each benchmark. Evaluation is performed on NVIDIA L40S GPUs.

Evaluation Metrics: To evaluate inference efficiency, we measure *Overall Throughput (tokens/s)*,

defined as the total number of generated tokens divided by the total inference time (prefill plus decoding). From this, we report the *Speedup*, calculated as the ratio of the model's throughput after layer skipping to the baseline throughput. A comprehensive breakdown of granular latency metrics, including prefill time and decode time per token, is detailed in Appendix A.3.

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*, described in Equation 4, to determine layer importance, and proposes skipping only during *prefill* stage. 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.

4.2 Evaluating different layer skipping metrics

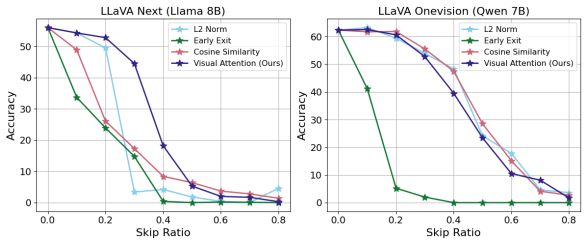


Figure 3: OCRBench accuracy for different layer skip metrics and skip ratios.

We compare different layer skipping metrics in Figure 3 using static skipping. We skip layers during the decode phase and plot the accuracy vs skip ratio on OCRBench for LLaVA with 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 demonstrate results from several other benchmarks in the following sections.

4.3 Results and Analysis

Evaluation on Image Captioning: Table 1 presents the performance of both our static and dynamic layer-skipping strategies across various image captioning benchmarks. We skip 15% to 20% of the decoder layers, achieving substantial overall throughput gains of up to 1.19× on Qwen3-VL and up to 1.16× on LLaVA architectures, with minimal degradation in output quality. Notably for Qwen3-VL, layer skipping consistently yields improved performance over baseline across all captioning tasks.

We observe that for the Qwen-based models (Qwen3-VL and LLaVA Onevision with Qwen2 backbone), dynamic skipping outperforms static skipping, achieving a higher average score across all captioning tasks. Conversely, for the LLaMA-based backbone (LLaVA-Next), static skipping proves marginally superior in preserving output quality on datasets such as NoCaps and CocoCaps. In terms of latency, our proposed framework explicitly targets the generative decoding phase. Because we do not skip layers during the context prefill stage, prefill latency remains unchanged. Consequently, the observed throughput improvements are driven entirely by the decoding phase, where latency reduction is directly proportional to the number of skipped layers. It is important to note that the overall throughput is primarily influenced by two factors: the length of the generated output, and the size of the input image (which dictates the baseline prefill latency). A detailed breakdown of throughput estimation, including prefill times, decode latencies, and generated sequence lengths is provided in Appendix A.3.

Comparison with existing methods: Table 2 compares our proposed static and dynamic layer-skipping strategies against existing layer skipping methods, namely ShortGPT and UIDL, across both image captioning and visual QnA tasks. Our framework consistently delivers a superior balance of efficiency and output quality, achieving overall throughput improvements of up to 1.31× for the LLaMA backbone and 1.21× for the Qwen backbone, while maintaining ROUGE_L scores remarkably close to the baseline.

Model	Dataset	Method	Skip Ratio	Bleu_1	Bleu_2	Bleu_3	Bleu_4	CIDEr	METEOR	ROUGE_L	Avg. Score	Speed -up
Qwen3-VL 8B	NoCaps		-	0.563	0.379	0.249	0.16	0.251	0.264	0.447	0.330	1x
		Static	0.15	0.537	0.359	0.234	0.15	0.31	0.253	0.441	0.326	**1.12**×
		Dynamic	0.15	0.566	0.384	0.254	0.164	0.328	0.26	0.454	**0.344**	1.02×
	Flickr30k		-	0.504	0.334	0.22	0.145	0.273	0.241	0.421	0.305	1×
		Static	0.15	0.482	0.322	0.212	0.139	0.313	0.233	0.419	**0.303**	**1.17**×
		Dynamic	0.15	0.452	0.303	0.201	0.1321	0.319	0.229	0.415	0.293	1.1×
	CocoCaps			0.478	0.307	0.192	0.117	0.269	0.252	0.419	0.291	1×
		Static	0.15	0.461	0.296	0.185	0.1132	0.242	0.332	0.416	0.292	**1.19**×
		Dynamic	0.15	0.464	0.299	0.187	0.1152	0.345	0.245	0.421	**0.297**	1.15×
LLaVA Onevision (Qwen2-7B)	NoCaps			0.747	0.590	0.454	0.341	0.877	0.288	0.553	0.550	1×
		Static	0.2	0.714	0.549	0.412	0.302	0.773	0.279	0.528	0.508	1.06×
		Dynamic	0.2	0.727	0.564	0.427	0.316	0.816	0.282	0.536	**0.524**	**1.14**×
	Flickr30k		-	0.771	0.591	0.434	0.313	0.802	0.252	0.536	0.528	1×
		Static	0.2	0.768	0.587	0.429	0.308	0.805	0.255	0.538	0.527	**1.11**×
		Dynamic	0.2	0.773	0.593	0.435	0.312	0.809	0.254	0.538	**0.531**	1.06×
	CocoCaps		-	0.816	0.669	0.533	0.423	1.405	0.313	0.611	0.681	1×
		Static	0.2	0.809	0.658	0.519	0.405	1.369	0.309	0.602	0.667	**1.14**×
		Dynamic	0.2	0.818	0.668	0.526	0.409	1.349	0.309	0.606	**0.669**	1.13×
	Detailcaps			0.646	0.418	0.261	0.165	0.114	0.203	0.302	0.301	1.11×
		Static	0.2	0.600	0.384	0.236	0.146	0.078	0.182	0.286	0.273	**1.16**×
		Dynamic	0.2	0.615	0.402	0.25	0.157	0.081	0.183	0.295	**0.283**	1.11×
LLaVA-Next (LLaMA3-8B)	NoCaps			0.868	0.737	0.595	0.465	1.146	0.299	0.615	0.675	1×
		Static	0.2	0.828	0.693	0.551	0.421	1.083	0.291	0.601	**0.638**	**1.16**×
		Dynamic	0.2	0.808	0.676	0.537	0.412	1.053	0.286	0.595	0.624	1.07×
	Flickr30k		-	0.765	0.587	0.429	0.308	0.767	0.246	0.532	0.519	1×
		Static	0.2	0.752	0.571	0.414	0.293	0.761	0.249	0.530	0.509	**1.13**×
		Dynamic	0.2	0.748	0.573	0.419	0.3	0.763	0.249	0.535	**0.512**	1.10×
	CocoCaps		-	0.818	0.673	0.534	0.419	1.380	0.305	0.607	0.677	1×
		Static	0.2	0.783	0.634	0.491	0.373	1.296	0.295	0.589	**0.637**	**1.14**×
		Dynamic	0.2	0.765	0.619	0.48	0.366	1.25	0.293	0.589	0.623	1.14×

Table 1: Performance comparison of static and dynamic layer skipping strategies on image captioning benchmarks. Baseline speedup is normalized to 1×. The highest average scores and speedups among the layer-skipping methods are highlighted in bold.

UIDL demonstrates a reasonable performance-efficiency trade-off for Qwen on captioning tasks; however, it suffers a severe accuracy degradation when applied to the LLaMA backbone. ShortGPT, conversely, performs poorly across both models and all tasks. Although ShortGPT restricts its layer skipping exclusively to the prefill stage, this early bypass proves detrimental. We hypothesize that because the skipped prefill layers never process and cache the necessary visual token representations, the subsequent decoding layers are unable to adequately leverage the image context.

Crucially, our evaluation finds that incorrect layer skipping can unintentionally trap the model in a loop, increasing the generation length by producing repetitive or nonsensical 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 Table 2. We attribute this failure to the selection criteria used by these baselines, such as cosine similarity and angular distance. These metrics incorrectly deem deeper LLaMA layers as less critical; however, bypassing them severely disrupts the model's capacity for coherent multimodal reasoning.

Comparison between static and dynamic skipping: In Table 2, we observe distinct task-dependent behaviors: static skipping generally better preserves performance on reasoning-heavy visual QnA tasks (e.g., OCRBench), whereas dynamic skipping yields the highest overall accuracy for descriptive captioning tasks. We hypothesize that this divergence stems from the nature of the tasks. Visual reasoning often requires the model to sequentially shift its focus across different parts of the image during the reasoning process. Consequently, the global attention patterns computed

Model	Dataset	Baseline	ShortGPT		UIDL		Ours (Static)		Ours (Dynamic)	
		ROUGE_L /Acc.	ROUGE_L /Acc.	Spd.	ROUGE_L /Acc.	Spd.	ROUGE_L /Acc.	Spd.	ROUGE_L /Acc.	Spd.
LLaVA-Next (LLaMA3-8B)	NoCaps	0.615	0.050	1.55×	0.253	1.58×	**0.601**	1.16×	0.595	1.07×
	Flickr30k	0.532	0.035	1.43×	0.245	1.42×	0.530	1.13×	**0.535**	1.10×
	CocoCaps	0.607	0.037	1.52×	0.228	1.54×	0.588	1.14×	**0.589**	1.14×
	OCRBench	0.560	0.002	1.08×	0.073	1.28×	**0.528**	1.31×	0.484	1.23×
	ChartQA	0.687	0.000	1.98×	0.127	2.05×	0.644	1.08×	**0.646**	1.14×
LLaVA Onevision (Qwen2-7B)	NoCaps	0.553	0.174	0.53×	**0.596**	0.97×	0.528	1.14×	0.536	1.06×
	Flickr30k	0.536	0.154	0.91×	0.536	1.15×	**0.538**	1.11×	**0.538**	1.06×
	CocoCaps	0.611	0.149	1.10×	0.575	1.14×	0.602	1.14×	**0.606**	1.13×
	DetailCaps	0.302	0.047	0.86×	0.289	1.17×	0.286	1.16×	**0.296**	1.11×
	OCRBench	0.623	0.049	0.63×	0.445	1.29×	**0.606**	1.21×	0.594	1.13×
	ChartQA	0.800	0.008	1.93×	0.387	1.01×	**0.799**	1.03×	0.786	1.02×
Qwen3-VL (8B)	OCRBench	0.827	0.763	1.26×	0.669	1.38×	**0.807**	1.1×	0.797	1.08×
	CocoCaps	0.419	0.3812	1.02×	**0.423**	1.13×	0.416	1.19×	0.421	1.15×

Table 2: Performance comparison with existing methods across datasets. Skip ratio is set to 20% for LLaVA models and 15% for Qwen3-VL. Baseline speedup is defined as 1×. We bold the highest ROUGE_L score among the layer-skipping methods to highlight quality retention.

during the initial prefill stage may not perfectly correlate with the localized attention patterns required deeper into the decoding phase.

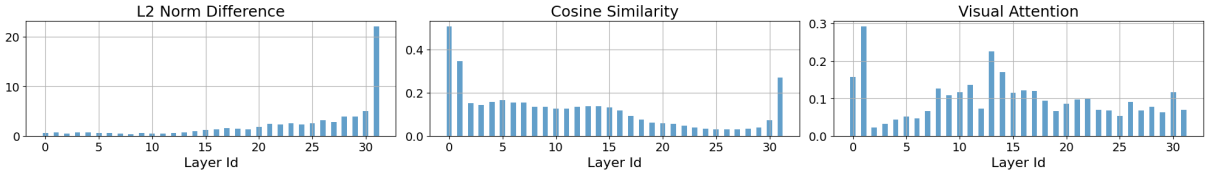
Regarding operational latency, the dynamic strategy incurs a minor computational overhead during prefill. Because intermediate attention weights are not available when using memory-efficient implementations like Scaled Dot-Product Attention (SDPA) or FlashAttention, computing the layerwise importance scores requires an auxiliary attention calculation during prefill. This slight increase in prefill latency explains why the throughput improvements for static skipping are higher than for dynamic skipping. Nevertheless, dynamic skipping operates entirely without prior calibration, enabling zero-shot deployment and robust adaptability to out-of-distribution images and new models.

Layer skipping for different model sizes: We present accuracy vs skip ratios on OCRBench using FlashVLM for LLaVA Onevision with Qwen-7B and Qwen-0.5B in Figure 5. This shows that bigger models are more resilient to layer skipping, suggesting higher layer redundancy in bigger models.

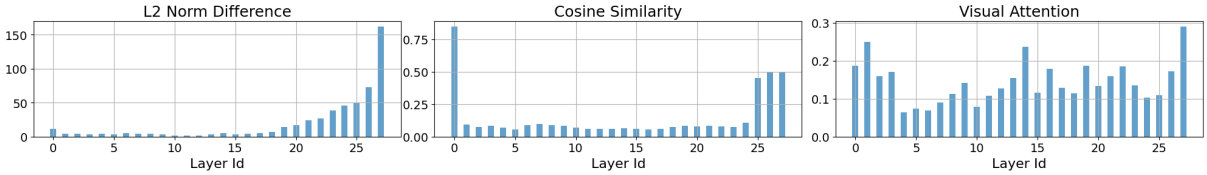
Layer Importance Score Visualization: We visualize the layerwise scores based on different metrics like L2 norm difference, cosine similarity and visual attention for LLaVA-Next (LLaMA3-8B) and LLaVA Onevision (Qwen2-7B) based on 500 samples from OCRBench (Liu et al., 2024c) in Figure 4a and 4b respectively. 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 later 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 4c. We observe that for the same model, similar attention patterns are observed across benchmarks. This explains how our static layer identification strategy, based on a subset of data samples for a particular task, can generalize effectively across different benchmarks.

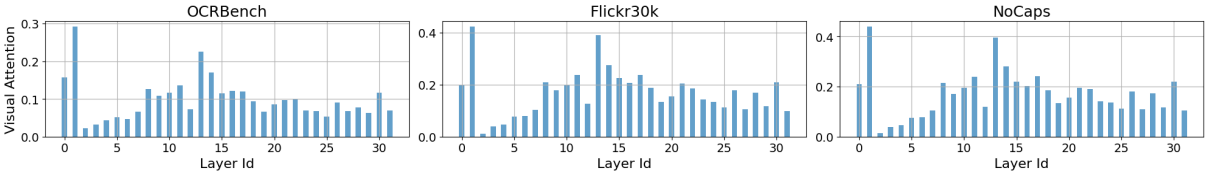
Comparison with token merging methods: For token merging approaches, we compare our method against ToMe (Token Merging) (Bolya et al., 2022) and PiToMe (Protect Informative Tokens before Merging) (Tran et al., 2024), which exploit token redundancy to increase throughput without requiring additional training. 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,



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

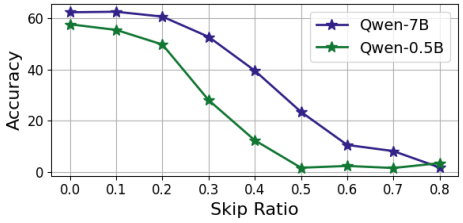


Figure 5: Evaluation across different model sizes

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)

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.

5 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 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. FlashVLM exhibits promising results, achieving notable inference speedups while maintaining performance across a wide range of tasks. In future, we aim to adapt this strategy for Chain-of-Thought (CoT) reasoning, where extended decoding lengths present a critical latency bottleneck.

Limitations

The current implementation of our layer-skipping strategy is specifically targeted at the decoding phase of VLM inference. Consequently, the strategy does not currently provide a reduction in prefill latency. The focus on decoding is increasingly justified by the shift toward complex reasoning tasks, such as Chain-of-Thought (CoT) reasoning. In these workloads, models generate extended sequences of tokens to solve intricate problems, making the decoding phase the primary bottleneck for total inference time. In such cases, the prefill latency becomes relatively negligible compared to the cumulative decode time. However, a more holistic approach that incorporates prefill-stage skipping remains an objective for future development. The adaptability of this strategy to complex reasoning tasks also remains to be explored.

References

Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. 2019. nocaps: novel object captioning at scale. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8948–8957.

Kazi Hasan Ibn Arif, JinYi Yoon, Dimitrios S Nikolopoulos, Hans Vandierendonck, Deepu John, and Bo Ji. 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, pages 1773–1781.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 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*, pages 19–35. Springer.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *Preprint*, arXiv:1504.00325.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hongyuan Dong, Jiawen Li, Bohong Wu, Jiacong Wang, Yuan Zhang, and Haoyuan Guo. 2024. Benchmarking and improving detail image caption. *arXiv preprint arXiv:2405.19092*.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *International Conference on Learning Representations*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, Bangkok, Thailand. Association for Computational Linguistics.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. 2025. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*.

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

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024.

Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. Llava-next: Improved reasoning, ocr, and world knowledge.

Shi Liu, Kecheng Zheng, and Wei Chen. 2024b. Paying more attention to image: A training-free method for alleviating hallucination in lvlms. In *European Conference on Computer Vision*, pages 125–140. Springer.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024c. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12).

Yaxin Luo, Gen Luo, Jiayi Ji, Yiyi Zhou, Xiaoshuai Sun, Zhiqiang Shen, and Rongrong Ji. 2025. $\gamma-$mod: Exploring mixture-of-depth adaptation for multimodal large language models. In *The Thirteenth International Conference on Learning Representations*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*.

Ahmed Masry, Do Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland. Association for Computational Linguistics.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, Vienna, Austria. Association for Computational Linguistics.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *Preprint*, arXiv:2404.02258.

Sreetama Sarkar, Yue Che, Alex Gavin, Peter A Beerel, and Souvik Kundu. 2025a. Mitigating hallucinations in vision-language models through image-guided head suppression. *arXiv preprint arXiv:2505.16411*.

Sreetama Sarkar, Gourav Datta, Souvik Kundu, Kai Zheng, Chirayata Bhattacharyya, and Peter A Beerel. 2025b. Maskvd: Region masking for efficient video object detection. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1955–1964. IEEE.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Chau Tran, Duy MH Nguyen, Manh-Duy Nguyen, TrungTin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Y Zou, Binh Nguyen, and Mathias Niepert. 2024. Accelerating transformers with spectrum-preserving token merging. *Advances in Neural Information Processing Systems*, 37:30772–30810.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *Preprint*, arXiv:2006.04768.

Shiwei Wu, Joya Chen, Kevin Qinghong Lin, Qimeng Wang, Yan Gao, Qianli Xu, Tong Xu, Yao Hu, Enhong Chen, and Mike Zheng Shou. 2024. VideoLLM-mod: Efficient video-language streaming with mixture-of-depths vision computation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 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.

Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Re. 2025a. LoLCATs: On low-rank linearizing of large language models. In *The Thirteenth International Conference on Learning Representations*.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and 1 others. 2025b. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, pages 18330–18341. Curran Associates, Inc.

A Appendix

A.1 Static Skipping Strategy

The detailed algorithm for identifying layer importance in static skipping is outlined 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}

A.2 Detailed Latency and Throughput Metrics

To provide a granular understanding of the computational savings achieved by our layer-skipping framework, we calculate and report the following detailed efficiency metrics:

- **Prefill Time / Sample:** The average time required to process the entire multimodal input sequence and generate the first output token.
- **Decode Time / Sample:** The average time required to generate all subsequent output tokens (excluding the first token) for a single sample.
- **Decode Time / Token:** The total decoding time divided by the total number of generated output tokens (excluding the first token).
- **Overall Throughput (tokens/s):** The total number of output tokens generated divided by the cumulative time spent in both the prefill and decode stages.
- **Speedup:** The relative efficiency gain, calculated as the overall throughput of the layer-skipping model divided by the baseline throughput.

A.3 Detailed Throughput Estimation for Captioning and QnA Tasks:

Evaluation of Image Captioning: 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 ∼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 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 4 and 5. 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.

Evaluation on Visual QnA: We present evaluation results for visual QnA tasks in Table 5. 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

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 4: 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.

the contrary, our approach shows promising results on these tasks with performance drops of 1.7% and 0.44% for OCRBench and ChartQA on Qwen. For LLaMA, FlashVLM degrades performance by 3.2% and 4.3% for OCRBench and ChartQA respectively. We obtain throughput improvements of up to 20.8% for OCRBench with a performance degradation of 1.7%. The throughput gains for ChartQA are limited due to their very short generation lengths. However, most real-world tasks like content summarization and UI navigation require substantially longer output, in which case our approach offers greater benefits.

A.4 Visualizing the output

The generated captions for the baseline and layer skipped model is visualized in Figure 6 We observe that (i) the semantic content is preserved, with both captions correctly describing the main entities (elephant, saddle, dirt path, river, greenery); (ii) the differences are largely stylistic or paraphrastic (e.g.,

Figure 6: Visualizing the generated captions with and without layer skipping on NoCaps.

```
Prediction: "A majestic elephant with a red seat saddle stands on a dirt path near a river, surrounded by lush greenery and trees."
Token Confidences: [('A', 0.8481), ('majestic', 0.7568), ('elephant', 0.9254), ('with', 0.9430), ('a', 0.9997), ('red', 0.5547), ('seat', 0.4668), ('saddl
e', 0.4422), ('stands', 0.9701), ('on', 0.3872), ('a', 0.6723), ('dirt', 0.96173), ('path', 0.99463), ('near', 0.43333), ('a', 0.95283), ('river', 0.972
7), (',', 0.9287), ('surrounded', 0.8326), ('by', 0.99997), ('lush', 0.9969), ('reen', 0.99699), ('ery', 0.7737), ('and', 0.8335282206535339), ('trees',
0.8119), ('.', 0.9109)]
```

```
Prediction: "A majestic elephant with a red seat and a saddle, standing on a dirt path near a riverbank in a lush green forest."
Token Confidences: [('A', 0.8481), ('majestic', 0.8572), ('elephant', 0.9752), ('with', 0.9653), ('a', 0.9995), ('red', 0.6411), ('seat', 0.4599), ('and',
0.6725), ('a', 0.9581), ('saddle', 0.3238), (',', 0.8086), ('standin', 0.9006), ('on', 0.2983), ('a', 0.7644), ('dirt', 0.8956), ('path', 0.9359), ('nea
r', 0.4199), ('a', 0.9639), ('river', 0.9697), ('bank', 0.7145), ('in', 0.37409), ('a', 0.8683), ('lush', 0.9511), ('reen', 0.8151), ('forest', 0.7885),
('.', 0.9352)]
```

Figure 7: Token-level confidence scores with and without layer skiping

"river" -> "riverbank", "surrounded by lush greenery" -> "in a lush green forest"); (iii) Token-level confidences, illustrated in Figure 7 remain comparable, particularly for key visual tokens such as "elephant", "dirt", "path", and "river", indicating that core visual grounding is unaffected.

This demonstrates that removing the identified low-importance layers does not significantly alter the model's ability to produce correct, grounded outputs, suggesting that these layers do not carry critical information required by later layers.

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%)**
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%)**

Table 5: 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.