Learning to Inference Adaptively for Multimodal Large Language Models

Zhuoyan Xu[†,*] Khoi Duc Nguyen[†,*] Preeti Mukherjee[§],
Somali Chaterji[§], Saurabh Bagchi[§], Yingyu Liang[†‡], Yin Li[†]
[†]University of Wisconsin-Madison, [§]Purdue University, [‡]The University of Hong Kong

Abstract

*Multimodal Large Language Models (MLLMs) have shown impressive capabilities in reasoning, yet come with substantial computational cost, limiting their deployment in resource-constraint environments. Despite some recent efforts on improving the efficiency of MLLMs, prior solutions yield models with static accuracy and latency footprint, and thus fall short in responding to varying runtime conditions, in particular changing resource availability (*e.g.*, contention due to the execution of other programs on the device). To bridge this gap, we introduce **AdaLLaVA**—an adaptive inference framework that learns to dynamically reconfigure operations in an MLLM during inference, accounting for the input data and a latency budget. We perform extensive experiments across multimodal benchmarks involving question-answering, reasoning and hallucination. Our results show that AdaLLaVA can adhere to input latency budget and achieve varying accuracy and latency trade-offs at runtime. Our project webpage with code release is at https://zhuoyan-xu.github.io/ada-llava/.*

1. Introduction

Large language models (LLMs) [3, 43] have been recently adapted to connect visual and text data. The resulting multimodal large language models (MLLMs), as exemplified by LLaVA [33, 34] and other recent works [2, 27, 29, 35, 58, 74], have shown impressive capabilities in visual reasoning, yet at the cost of significant computational cost. Several recent efforts seek to improve the efficiency of MLLMs by considering lightweight architectures, mixture of experts, or token selection techniques [32, 50, 70, 74]. A common characteristic of these methods is that they yield models with static accuracy and latency footprint during inference.

We argue that MLLMs with fixed computational footprint are insufficient for real-world deployment. Consider the example of deploying an MLLM on a server farm. Different requests may have distinct latency requirements, *e.g.*, requests from a mobile application, which requires instant feedback to an user vs. those from a recommendation system, which performs updates less frequently and thus can tolerate a higher latency. Further, the available computing resources may vary at any given point in time, as the overall loads of the system fluctuate. Similarly, when deployed on an edge device, the latency budget often remains constant, yet the computing resources may vary due to contention produced by other concurrent on-device programs.

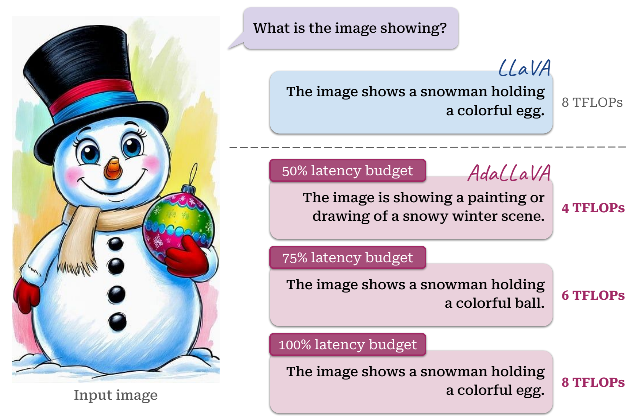


Figure 1. Given an image-query pair and latency constraints, AdaLLaVA learns to generate appropriate responses while adapting to varying computational budgets.

Different from prior approaches, we propose to address latency-aware adaptive inference for MLLMs, aiming to dynamically adjust a model's computational load based on input content and a specified latency budget. This problem is of both conceptual interest and practical significance. Our key insight is that a modern MLLM can be conceptualized as a collection of shallower models, and choosing among these models allows for dynamic reconfiguration during inference. For example, prior works have shown that Transformer blocks in an LLM and some attention heads within these blocks can be bypassed with minor impact on accuracy [6, 8, 56] and reduced latency. Thus, strategically selecting these operations during inference leads to a set of models with shared parameters but distinct accuracy-latency tradeoffs, thereby enabling the MLLM to flexibly respond to varying latency budgets.

To this end, we present **AdaLLaVA**, a learning based

[*]Equal contribution

framework for adaptive inference in MLLMs. As shown in Fig. 1, given an input image, a text query and a latency budget, **AdaLLaVA** enables an MLLM to answer the query about the image while adhering to the latency budget— a capability unattainable with the base MLLM. Key to AdaLLaVA lies in a learned scheduler that dynamically generates an execution plan, selecting a subset of operations within the MLLM based on the input content and a specified latency budget. This execution plan ensures that inference is performed within the given latency constraint while maximizing expected accuracy. To enable effective learning of the scheduler, we introduce a probabilistic formulation in tandem with a dedicated sampling strategy to account for latency constraints at training time.

To evaluate AdaLLaVA, We conduct extensive experiments. Our results demonstrate that AdaLLaVA can achieve a range of accuracy-latency tradeoffs at runtime. AdaLLaVA maintains comparable performance to base MLLMs across several benchmarks while operating with higher efficiency. Further, AdaLLaVA exhibits strong adaptability to different latency budgets, effectively trading accuracy for speed during inference, particularly in extremely latency-constrained settings. Importantly, in all cases, AdaLLaVA adheres to latency budgets. Additionally, AdaLLaVA can be further integrated with token selection techniques to further enhance efficiency, and shows content-aware adaptation by generating execution solutions catered for specific input samples.

Our key **contributions** are three folds.

- We present AdaLLaVA, a novel adaptive inference framework for MLLM. Our method for the first time enables dynamic execution of MLLMs based on a latency budget and the input content at inference time.
- Our key technical innovation lies in (1) the design of a latency-aware scheduler, which reconfigures a base MLLM model during inference; and (2) a probabilistic modeling approach, which allows for the incorporation of hard latency constraints during MLLM training.
- Through extensive experiments, we demonstrate that AdaLLaVA can adapt to a range of latency requirements while preserving the performance of the base model, and that AdaLLaVA can be integrated with token selection techniques to further enhance efficiency.

2. Related Work

Multimodal Large Language Models (MLLMs). With the success of LLMs, increasing research focus on extends LLMs from pure text modality to other modalities such as image [34], video [27], and audio [26]. Such development leads to the emergence of MLLMs, often involving combining vision encoders with existing LLMs. Flamingo [2] inserts gated cross-attention dense blocks between vision encoder and LLMs, align vision and language modality. BLIP2 [29] introduce Q-former with two-stage pretraining, bridge frozen image encoders and LLMs to enable visual instruction capability. LLaVA [33, 34] and MiniGPT-4 [74] use simple MLP to connect vision embedding space and text token space and show state-of-art performance on a variety of tasks. Our work builds on these developments and aims to enable adaptive inference of MLLMs.

Adaptive Inference. Adaptive inference refers to the capability in which the computational complexity of making predictions is dynamically adjusted based on the input data, latency budget, or desired accuracy levels [16]. Early works focus on the selection of hand-crafted features in multi-stage prediction pipelines [15, 24, 64]. More recent works have extended these ideas to deep models. For convolutional networks, methods have been developed to downsample the input, skip layers or exist early during inference [4, 12, 19, 23, 28, 41, 59, 63]. For vision transformers, various approaches have been proposed to enhance efficiency, such as selecting different patches of images [45, 47, 60], and using different attention heads and blocks [25, 40]. Similar ideas have also been explored for LLMs, where models selectively process tokens [48] or execute a subset of the operations [11, 49] during inference.

Our approach builds upon these ideas by dynamically selecting a subset of model components during inference. Unlike existing methods, our approach specifically targets the inference of MLLMs under latency constraints, predicting feasible execution plans tailored for each input while adhering to varying budget budgets.

Efficient Inference for MLLMs. MLLMs face a major challenge in deployment, due to their high computational costs during inference. Several recent works design lightweight model architectures to reduce the costs. Examples include Phi-2 [22], Tinygpt-v [71] and LLaVA-ϕ [75]. Vary-toy [62] enhanced performance through specialized vision vocabulary in smaller models. TinyLLaVA [73] and LLaVA-OneVision [27] learn small-scale models with better training data and pipeline. MoE-LLaVA [32] and LLaVA-MoD [53] improve efficiency by incorporating mixture-of-experts architectures and parameter sparsity techniques. Another line of research investigates the selection of input tokens to improve efficiency. An input image or video can lead to a large number of vision tokens. To address this, MADTP [7] and LLaVA-PruMerge [50] introduce token pruning and merging technique to reduce the tokens counts. Pham et al. [46] propose to selectively disabling attention mechanisms for visual tokens in MLLMs.

While our approach also aims to improve the efficiency of MLLMs, it focuses dynamically adjusting an MLLM to fit varying latency budget during inference. This makes our approach orthogonal to prior efforts centered on developing inherently efficient MLLMs. Through our experiments, we will demonstrate that our approach is compatible with smaller models and integrates seamlessly with existing token-pruning techniques *e.g.*, LLaVA-PruMerge [50].

3. Adaptive Inference of MLLMs

We propose **AdaLLaVA**, an adaptive inference framework for MLLMs. Given a latency budget and a multimodal sample at inference time, our framework employs a scheduler learned from data to dynamically reconfigure the execution of MLLMs. Importantly, the scheduler strategically selects a subset of operations to execute, catered for the input budget and content. In doing so, our approach ensures that the inference adheres to the latency constraint while preserving model accuracy. Fig. 2 (a) provides an overview of our framework, where our designed scheduler takes input from both multimodal sample and latency budget and outputs an execution plan tailored to that specific input.

In what follows, we introduce the background on MLLMs (Sec. 3.1), outline our key idea for scheduling MLLMs (Sec. 3.2), present our approach for training and inference with the scheduler (Sec. 3.3), and further describe the specifics of our solution (Sec. 3.4).

3.1. Preliminaries: MLLMs

A MLLM takes an image (or video) $\mathbf{X}^v$ and a text query $\mathbf{X}^q = \{x^q\}$ as its input, and generates an answer $\mathbf{X}^a = \{x^a\}$ in text format. Specifically, $\mathbf{X}^v$ is first encoded by a visual encoder (including the vision backbone and its projector) $h_v(\cdot)$ into a set of visual tokens $\{\mathbf{z}^v \in \mathbb{R}^d\}$. Similarly, $\mathbf{X}^q$ is processed by a text encoder $h_t(\cdot)$, which embeds words x^q into text tokens $\{\mathbf{z}^q \in \mathbb{R}^d\}$ with $\mathbf{z}^q = h_t(x^q)$. Theses tokens are further combined into $\{\mathbf{z}^{v|q}\} = [\{\mathbf{z}^v\}, \{\mathbf{z}^q\}]$, and processed by an LLM $f(\cdot)$, which decodes $\mathbf{X}^a$ in an autoregressive manner

$$f\left(\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right]; \theta\right) \rightarrow x^a_i, \tag{1}$$

where $\{\mathbf{z}^a_{<i}\}$ are text tokens from previously generated answer $x^a_{<i}$ *i.e.* $\mathbf{z}^a = h_t(x^a)$, and θ denotes LLM parameters.

For the rest part of the paper, we will primarily consider the learning of LLM parameters θ — the major portion of parameters within the MLLM. Yet we note that learning encoder parameters (in $h_v(\cdot)$ and $h_t(\cdot)$) can be done similarly.

3.2. Reconfiguring and Scheduling MLLMs

Dynamic reconfiguration. Our key insight is that MLLM can be conceptualized as a collection of shallower models with shared parameters yet distinct accuracy-latency tradeoffs, enabling dynamically reconfiguration during inference. To this end, we propose to equip the LLM $f(\cdot)$ with K tunable binary switches $\mathbf{s} \in (0,1)^K$, where $\mathbf{s}$ determines the execution of individual operations at runtime, such as a Transformer block or an attention head. The state of each operation will be controlled by a switch, enabling (1) or disabling (0). We defer the choice of these operations and the design of these switches to our model instantiation. Here, we first focus on the key concept of LLM decoding, which is given by

$$f\left(\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right], \mathbf{s}; \theta\right) \rightarrow x^a_i. \tag{2}$$

Specifically, $f(\cdot)$ now takes the switches $\mathbf{s}$ as an additional input, and only executes a subset of operations when generating its output. It is worth noting that the switches $\mathbf{s}$ does not depend on the decoding step i, *i.e.*, a fixed set of operations are used to decode all tokens in the output, though this set may varying across different inputs.

Scheduler. The crux of our method lies in learning a scheduler $g(\cdot)$ that controls the execution of $f(\cdot)$ during inference. The scheduler $g(\cdot)$ predicts a configuration of the switches $\mathbf{s}$ based on input tokens $\{\mathbf{z}^{v|q}\}$ and an inference latency budget l. This is written as

$$g\left(\{\mathbf{z}^{v|q}\}, l; \phi\right) \rightarrow \mathbf{s}, \tag{3}$$

where ϕ denotes learnable parameters of the scheduler $g(\cdot)$.

The goal of $g(\cdot)$ is to determine an execution plan that meets the latency requirement while preserving the accuracy. This amounts to estimating the solution to the following combinatorial optimization problem *for each input*.

$$\begin{aligned} \min_{\mathbf{s}} \quad & -\Sigma_i \log p\left(x^a_i = f\left(\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right], \mathbf{s}; \theta\right)\right) \\ \text{s.t.} \quad & \text{Latency}\left(f\left(\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right], \mathbf{s}; \theta\right)\right) \leq l. \end{aligned} \tag{4}$$

Here the objective is to minimize the negative log likelihood of decoded text — the standard loss used for training LLMs, and the constraint states that the latency of executing the model must fall under the budget.

3.3. Learning to Schedule Probabilistically

Learning the scheduler $g(\cdot)$ presents a major challenge. While it is tempting to pursue a fully supervised approach, where $g(\cdot)$ is trained to exactly predict the solution to Eq. 4, doing so requires solving the optimization for each sample at every iteration. Even with a small number of switches, this is prohibitively expensive.

Deterministic modeling. A possible solution is to solve a relaxed version of the constrained optimization at training time. We initially explored this solution, where we let the scheduler predict the hard execution plans on binary switches s and attribute latency violation into part of the objective, leading to the following loss

$$\arg\min_{\theta,\phi} \; -\Sigma_i \log p\left(x^a_i = f(\cdot)\right) + \lambda \|\text{Latency}(f(\cdot)) - l\|_2^2,$$

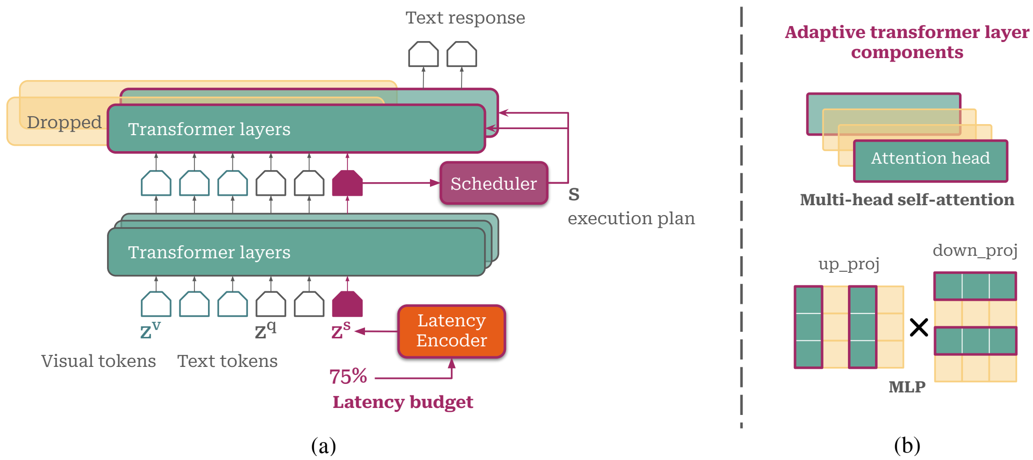


Figure 2. Overview of **AdaLLaVA**. **(a)** learning based latency encoder and and scheduler. The encoder will embed latency budget into an additional latency token, This token's embeddings are extracted from specific intermediate layers and fed to the scheduler, determining execution plans for components in subsequent layers. These plans can control either complete layers or specific subsets within layers. **(b)** Within each layer, our design focuses on two primary components: attention heads and MLP neurons, specifically their activation values. The control over MLP neurons can be achieved using a subset of the weight matrix.

where λ can be considered as the Lagrangian multiplier. Here the execution of $f(\cdot)$ depends on the output of the scheduler of $g(\cdot)$, allowing us to jointly optimize the LLM $f(\cdot)$ and the scheduler $g(\cdot)$.

This deterministic approach is further described in the supplement. Empirically, we found that this method fails to enforce a strict latency constraint on the scheduler and often produces suboptimal execution plans that exceed latency limits or under-utilize the available resources. We demonstrate this limitation through experimental results in Sec. 4.4 and present further discussion in our supplement.

Probabilistic modeling. To address this challenge, we propose a probabilistic model to relax the constraints, avoiding directly solving Eq. 4 while stabilizing the joint training of the LLM and the scheduler. The key idea is to impose a distribution over the choice of the switches $\mathbf{s}$, in lieu of making a hard decision. Specifically, we design the scheduler to predict a probabilistic distribution

$$g\left(\{\mathbf{z}^{v|q}\}, l; \phi\right) \approx p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right). \tag{5}$$

With minor abuse of the notation, $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)$ is the probability of triggering binary switches $\mathbf{s}$ given the input $\{\mathbf{z}^{v|q}\}$, latency budget l, and the scheduler parameters ϕ. Ideally, $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi) = 0$ if the execution latency exceed the budget, and can be positive otherwise.

We now re-formulate the inference of MLLM as sampling from the following hierarchical distribution.

$$\begin{aligned} \mathbf{s} &\sim p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right), \\ x_i^a &\sim p\left(x_i^a|\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right], \mathbf{s}, \theta\right). \end{aligned} \tag{6}$$

We produce execution plans through such conditional sampling strategy, we first let scheduler g output probability of keep/drop each switch, approximating the distribution $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)$, we then sample execution plans $\mathbf{s}$ based on the distribution without violating the latency constraint.

In actual MLLMs inference, given an input and a latency budget, we first sample an configuration plan from the scheduler, and then execute this plan to further generate answers from the LLM.

Training loss. This formulation allows us to directly optimize the following loss function for training.

$$\underset{\theta,\phi}{\arg\min}\ \mathbb{E}_{\mathcal{D}}\left[-\log p\left(x_i^a|\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right], l, \theta, \phi\right)\right],$$

where $\mathcal{D}$ is the data distribution approximated by the training set $(\mathbf{X}^v, \mathbf{X}^q, \mathbf{X}^a, l) \sim \mathcal{D}$. By marginalizing $\mathbf{s}$, we have

$$\begin{aligned} &p(x_i^a|[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}], l, \theta, \phi)) = \\ &\quad \mathbb{E}_{p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)}\left[p(x_i^a|[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}], \mathbf{s}, \theta)\right], \end{aligned}$$

Thus, the loss function is transformed into

$$\underset{\theta,\phi}{\arg\min}\ \mathbb{E}_{\mathcal{D},\mathbf{s}\sim p(\mathbf{s}|\cdot)}\left[-\log p\left(x_i^a|\left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}^a_{<i}\}\right], \mathbf{s}, \theta\right)\right],$$

where $p\left(\mathbf{s}|\cdot\right) = p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right)$.

More concretely, this loss can be computed by (1) sampling an input data point and a latency budget from the training data, (2) sampling an execution plan, *i.e.* a configuration of the switches, from the scheduler output, so that it satisfy the latency budget (3) execute the plan and generate the answer, and (4) evaluate standard negative log likelihood of the decoded text. Optimizing this loss additionally requires back propagation through the sampling process $\mathbf{s} \sim p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right)$, which we approximate using the Gumbel Softmax trick [21, 39].

Adaptive inference. During inference, the scheduler outputs the probability $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)$ of choosing individual switches $\mathbf{s}$, based on the input $\{\mathbf{z}^{v|q}\}$ and the latency budget l. In theory, the inference requires marginalizing this distribution for decoding the answer x_i^a at each step. In practice, we approximate the inference by simply plugging the best execution plan from the scheduler for every step. This approximation bypasses the expectation term, and thus remains highly efficient. We have empirically verified its effectiveness. Formally, this approximation is given by

$$\begin{aligned} x_i^a &= \arg\max_{x_i^a} \, \mathbb{E}_{\mathbf{s}\sim p(\mathbf{s}|\cdot)} \left[-\log p\left(x_i^a | \left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}_{<i}^a\}\right], \mathbf{s}, \theta\right)\right] \\ &\approx \arg\max_{x_i^a} \, p\left(x_i^a | \left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}_{<i}^a\}\right], \mathbf{s}^*, \theta\right), \end{aligned}$$

where $\mathbf{s}^* = \arg\max_{\mathbf{s}} p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right)$. Note that model parameters θ and ϕ are now fixed.

3.4. Model Instantiation

The design of tunable switches. We attach binary switches to the LLM, which accounts for the majority of computational costs. We implement two distinct approaches to select operations at inference time using binary switches.

- `AdaLLaVA-L` (layer-level): In this design, binary switches are attached to entire transformer blocks. When a switch is off, the corresponding block is bypassed through its residual connection, becoming an identity mapping. The execution plan determines whether each layer is computed or bypassed (see Fig. 2(a)).
- `AdaLLaVA-H` (head/neuron-level): This design applies binary switches to individual components within layers, including individual attention heads within attention modules and specific neuron activation values in MLP layers, similar to selective dropout for these components (see Fig. 2(b)).

To ensure stable and consistent model performance, we fixed the first half of transformer blocks, applying dynamic execution plans exclusively to the latter half.

Sharing parameters between the LLM and scheduler. A key design choice is to reuse the parameters and operations from LLM $f(\cdot)$ for the scheduler $g(\cdot)$. Specifically, we design a latency encoder that converts the latency budget into a token embedding, which is then concatenated with the original input sequence before feeding into LLM layers. The latency token's representations are captured from intermediate layers and passed to a lightweight scheduler, which outputs the execution plan. Notably, the lower half of the MLLM's layers serve two purposes: they simultaneously process regular MLLM tasks and learn latency-aware resource allocation based on both content and budget constraints. This design is depicted in Fig. 2 (a).

Approximating $p\left(\mathbf{s}|\cdot\right)$. To model the switch configuration distribution $p\left(\mathbf{s}|\mathbf{z}^{v|q}, l, \phi\right)$ under latency constraints, our scheduler outputs probability scores for each switch. We employ the conditional sampling strategy: switches are sampled one at a time without replacement until reaching the specified latency budget. The resulting execution plan is then implemented, and we optimize the model by minimizing our loss.

Implementation details. We adopt the architecture of LLaVA [33] and integrate the scheduler into its LLM (see Fig. 2). Our latency encoder is implemented using sine and cosine functions, and the scheduler is a simple linear layer (randomly initialized) that maps the latency token to the logits for the switches. The scheduler takes the latency token processed by the first half of the transformer layers and generates the execution plan for the second half. This results in a model with latency ranging from 50% to 100% of the original LLaVA model.

To quantify model latency, we adopt a computational complexity-based approach, using FLOPs (floating point operations) as our primary metric. This provides a hardware-agnostic measure of computational cost that directly correlates with actual runtime performance. Our FLOPs calculation methodology mainly follows the standardized procedures established in [72].

4. Experiments and Results

We now present our experiments and results. We introdue our setup (Sec. 4.1), present our main results (Sec. 4.2), provide further analysis for the scheduler (Sec. 4.3), and conduct ablation studies (Sec. 4.4).

4.1. Experimental Setup

Training details. Instead of following LLaVA's two-stage training procedure, we focus on jointly finetuning its LLM and training the scheduler on visual instruction data, while keeping the vision encoder frozen. We initialize our model with the pretrained LLaVA-1.5 checkpoint. During finetuning, each training sample is paired with a randomly generated latency requirement ranging from 0.5 to 1.0 (as we only operate on the top half of the layers in LLM). We set learning rate to 10^{-5} for the LLM and 10^{-4} for the scheduler, while keeping other training hyperparameters consistent with the original LLaVA stage-2 finetuning protocol.

Benchmarks. We conduct comprehensive evaluations across multiple visual understanding benchmarks, including VQAv2 [14], ScienceQA [38], TextVQA [54], MME [13], and MMBench [37]. We also evaluate on hallucination benchmarks such as POPE [30]. For the TextVQA evaluation, we specifically focused on the image-based subset,

Method	LLM	Percentage (%)	FLOPs (T)	VQA^{v2}	SQA^{I}	VQA^{T}	POPE	MME	MMB
BLIP-2	Vicuna-13B	100	-	41.0	61	42.5	85.3	1293.8	-
InstructBLIP	Vicuna-7B	100	-	-	60.5	50.1	-	-	36
InstructBLIP	Vicuna-13B	100	-	-	63.1	50.7	78.9	1212.8	-
Shikra	Vicuna-13B	100	-	77.4	-	-	-	-	58.8
IDEFICS-9B	LLaMA-7B	100	-	50.9	-	25.9	-	-	48.2
IDEFICS-80B	LLaMA-65B	100	-	60.0	-	30.9	-	-	54.5
Qwen-VL	Qwen-7B	100	-	78.8	67.1	63.8	-	-	38.2
Qwen-VL-Chat	Qwen-7B	100	-	78.2	68.2	61.5	-	1487.5	60.6
LLaVA-1.5	Vicuna-7B	100	9.3	78.5	66.8	58.2	85.9	1510.7	64.3
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-7B	85	8.1	77.3	68.1	53.9	86.4	1505.3	64.3
LLaVA-1.5 w/ `AdaLLaVA-H`	Vicuna-7B	85	8.1	76.9	67.9	54.5	86.0	1502.6	62.5
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-7B	60	5.8	73.5	66.6	45.4	85.2	1490.2	61.9
LLaVA-1.5 w/ `AdaLLaVA-H`	Vicuna-7B	60	5.8	72.1	67.0	45.6	86.6	1480.7	61.8
LLaVA-1.5	Vicuna-13B	100	18.2	80.0	71.6	61.3	85.9	1531.3	67.7
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-13B	85	15.9	79.1	72.6	58.1	86.1	1519.3	68.3
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-13B	60	11.3	77.1	71.9	54.7	86.9	1517.1	68.5
Prumerge	Vicuna-7B	100	0.91	72.0	68.5	56.0	76.3	1350.3	60.9
Prumerge w/ `AdaLLaVA-L`	Vicuna-7B	85	0.77	68.6	68.6	51.8	74.0	1375.7	57.6
Prumerge w/ `AdaLLaVA-L`	Vicuna-7B	60	0.54	65.6	68.4	44.1	75.6	1351.5	55.6

Table 1. Results of MLLMs on six benchmarks. Our AdaLLaVA can be applied to LLaVA 1.5 with different size of LLM with different design of switches. Percentage (%): The input latency requirement. `AdaLLaVA-L`: switches on selecting different transformer blocks. `AdaLLaVA-H`: switches on select different attention heads and MLP activations. VQA^{v2}: VQAv2 set. SQA^{I}: ScienceQA set. VQA^{T}: TextVQA set. Prumerge: LLaVA 1.5 with PruMerge.

where each question is paired with corresponding image content. For each benchmark, we report the official metrics on the same dataset splits as in LLaVA-1.5. In each evaluation of AdaLLaVA, the same latency requirement (from 0.5 to 1.0) is applied across all sample in the dataset.

4.2. Main Results

Setup. Following LLaVa-1.5, we consider two model sizes, *i.e*., 7B and 13B. We evaluate AdaLLaVA with two different designs: (a) AdaLLaVA-L for selecting Transformer blocks; and (b) AdaLLaVA-H for selecting attention heads and MLP activations. To demonstrate the efficacy of AdaLLaVA, we consider two latency budgets: 60% and 85%. Additionally, we report the FLOPs during the prefill stage for an efficiency comparison.

Results and discussion. Our main results across six benchmarks are summarized in Table 1. AdaLLaVA framework shows comparable performance while achieving efficiency improvements across multiple benchmarks. When applied to LLaVA-1.5 with Vicuna-7B, AdaLLaVA-L maintains similar performance with only 85% compute requirement. For instance, our method achieves 64.3 on MMB, same as full model. We found on our method surpass the full model performance on certain benchmarks, such as ScienceQA (68.1 vs 66.8), and POPE (86.4 vs 85.9). Similar results are observed with AdaLLaVA-H, which focuses on selecting attention heads and MLP activations. Given 60% compute requirement, our AdaLLaVA-H still maintains strong performance comparing to full model on certain benchmarks, such as ScienceQA (67.0 vs 66.8) and POPE (86.6 vs 85.9). The effectiveness of our approach also scales to larger models, as demonstrated by the results with Vicuna-13B backbone. Notably, in some cases, our method outperforms the baseline while using only 60% computational resources, as seen in ScienceQA (71.9 vs 71.6), POPE (86.9 vs 85.9) and MMB (68.5 vs 67.7).

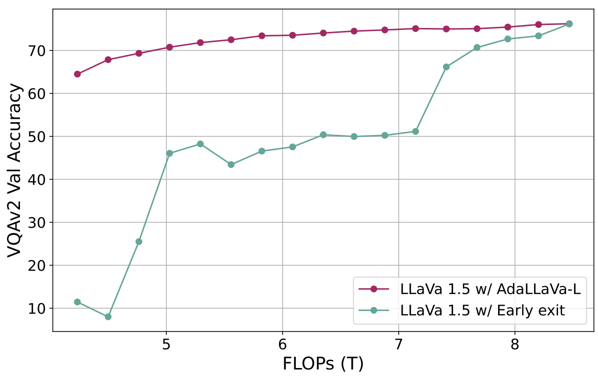
Integration with Token Selection Techniques. Our method demonstrates strong compatibility with other efficiency techniques, such as token pruning method. As shown in Tab. 1, when integrated with LLaVA-PruMerge, AdaLLaVA maintains competitive performance across multiple benchmarks while significantly reducing computational costs. Notably, AdaLLaVA-L with PruMerge achieves 68.6 accuracy on ScienceQA while using only 0.77T FLOPs.

Overall, these results demonstrate that AdaLlava can effectively maintain model performance while significantly reducing computational requirements, offering a practical solution for deploying large multimodal models under varying resource constraints.

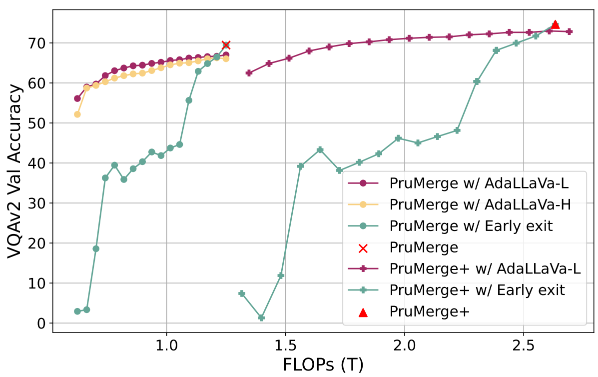
4.3. Latency- and Content- Aware Scheduling

Latency awareness. We perform a comprehensive evaluation of AdaLLaVA under varying latency constraints. We conduct extensive experiments on VQAv2 benchmark to demonstrate the adaptive ability of our approach.

To make comprehensive comparison, we also implemented a simple baseline that involves naively truncating the original model to meet latency constraints. We remove layers from the model with greedy manner, starting from



(a) Results on LLaVA 1.5-7b.



(b) Results on LLaVA 1.5-7b with PruMerge and PruMerge+.

Figure 3. Results on VQAv2 benchmark across latency budgets (FLOPs). AdaLLaVA-L/H: our methods. Early Exit: Naive truncation method. We denote performance of full model as a single point.

the top, until the desired latency target is achieved. We note this naive method as early exit.

We show our results on LLaVA 1.5 in Fig. 3a. When varying latency requirements from 0.5 to 1.0, our AdaLLaVA-L demonstrate smooth and consistent performance scaling as the available computational resources (FLOPs) increase. The curves show a clear upward trend, validating that our scheduler can make decisions in selecting model components - as more computational budget becomes available, it progressively activates more components, leading to better performance and eventually matching full LLaVa 1.5 performance. We also notice that the naive early exit baseline performs significantly worse than our adaptive approach. This highlights the importance of latency-aware model design, as the original model lacks latency awareness and fail to adapt to flexible computational constraints.

To demonstrate the versatility of AdaLLaVA, we further integrate it with other efficient inference methods on LLaVA, namely, PruMerge and PruMerge+ [50], which reduce computation by pruning majority of visual tokens. We report results in Fig. 3b. We observe that PruMerge+

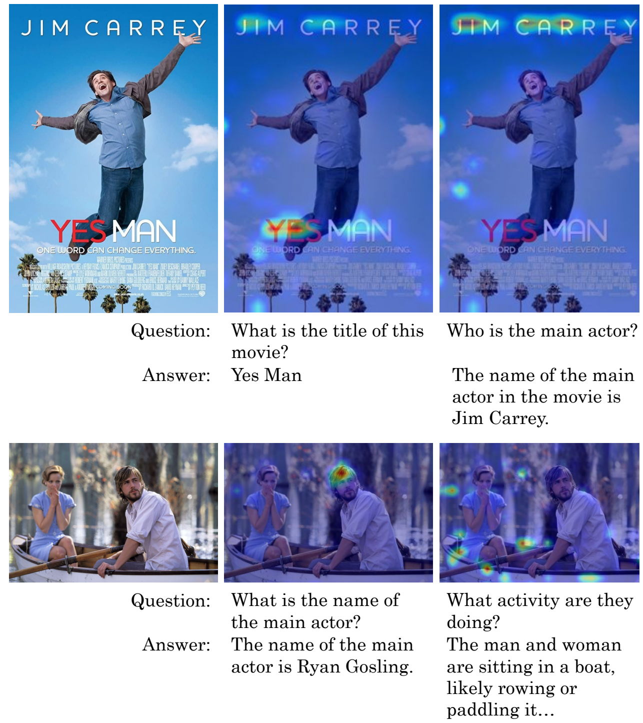


Figure 4. The key-query attention scores between latency token and visual tokens. The latency input is 1.0 in these examples.

achieves higher accuracy but at the cost of increased FLOPs. In both case AdaLLaVA demonstrates efficient performance scaling with computational resources, consistently outperforming the early exit baseline. The results show that our adaptive approach is complementary to existing efficiency methods, achieving strong performance from as low as 1.0T FLOPs to matching the accuracy of PruMerge+ at 2.5T FLOPs.

Content awareness. The design of AdaLLaVA enables execution plans to adapt dynamically to different input content. Fig. 4 shows the key-query attention scores of the latency token and the input visual tokens with different text questions. The attention scores are taken right before the latency token is fed into the scheduler. As shown in Fig. 4, our scheduler demonstrates content-aware attention scores across different queries. For the top poster, attention concentrates on 'YesMan' for the title question but shifts to the name elements for actor identification question. Similarly, in bottom picture, attention spreads across the scene elements when describing activities but focuses specifically on the character for actor identification question. This shows our model's ability to dynamically adjust its computational focus based on the query type.

In Fig. 5, we directly investigate the execution plans given different input content. When processing similar queries (e.g., asking about creators) for different artworks, the scheduler generates distinct execution patterns, demon-

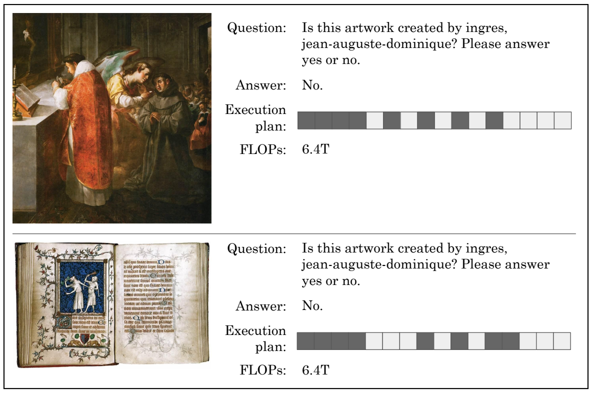


Figure 5. The execution plans generated by scheduler given different visual input. The execution plans represents either enable (shallow color) or disable (deep color) for the 16 to 32 layers. The latency input is 75%.

strating its ability to allocate resources based on specific visual content. The various in attention score and execution plans show that our scheduler learns to make decisions depend on visual and text input rather than applying a fixed, content-agnostic strategy.

4.4. Ablation Study

We now conduct ablation study, exploring different design choices. Due to space limit, we only present results with LLaVA 1.5-7b Model on VQAv2 dataset benchmark.

Design of the switches. We explore the performance of design of tunable switches, namely AdaLLaVA-L and AdaLLaVA-H (detailed in Sec. 3.4). AdaLLaVA-L allows adaptivity to latency requirements while not changing too much the well-trained LLM. On the other hand, AdaLLaVA-H offers better flexibility to latency input but requires significant change to the model architecture. Fig. 6 shows the performance scaling of our two switching strategies on VQAv2. While AdaLLaVA-L achieves slightly better accuracy across most computational budgets, AdaLLaVA-H demonstrates finer-grained control over the accuracy-latency trade-off. This is evident from the smoother curve of AdaLLaVa-H, which can be attributed to its head/neuron-level switches providing more granular control over computational resources compared to the layer-level switches. This flexibility allows AdaLLaVa-H to accommodate a wider range of latency budgets, though at a slight cost of lower peak performance.

Probabilistic vs. deterministic modeling of latency constraints. We investigate two approaches to the scheduler design: deterministic and probabilistic (see Sec. 3.3). The deterministic scheduler directly outputs execution plans and combines latency and language model losses. For our main experiments, we adopt the probabilistic approach with conditional sampling (detailed in Section 3.4). Here we compare these two modeling paradigms, evaluating their performance across different latency constraint.

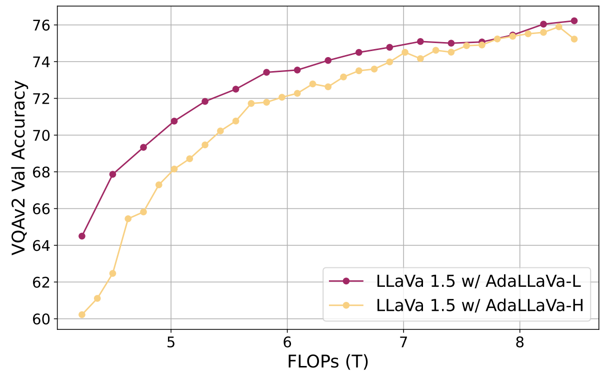


Figure 6. Results comparing two choices of switches.

	AdaLLaVa-L		**Deterministic scheduler**	
Latency budget	**Accuracy**	**FLOPs** (T)	**Accuracy**	**FLOPs** (T)
0.5	64.5	4.2	33.5	4.2
0.56	69.3	4.8	62.9	4.5
0.63	71.8	5.3	69.1	5.0
0.69	73.4	5.8	70.2	5.3
0.75	74.1	6.3	72.4	6.0
0.81	74.8	6.9	75.4	8.2
0.88	75.0	7.4	75.5	8.5
0.94	75.4	7.9	75.7	8.5
1.0	76.2	8.5	75.0	8.5

Table 2. Results of AdaLLaVA-L and deterministic scheduler across latency budget. Red values indicate computation violation.

As illustrated in Table 2, AdaLLaVA-L demonstrates superior adaptability across different latency budgets compared to the deterministic approach. We notice deterministic approach has performance drop given low latency budget due to under-utilization, and violates the higher latency input. This suggests AdaLLaVA-L achieves better resource efficiency while maintaining higher accuracy, particularly at stricter latency constraints.

5. Conclusion

In this paper, we introduced AdaLLaVA, a novel adaptive inference framework designed to address the critical challenge of deploying MLLMs in resource-constrained environments. Our approach features a lightweight, learning-based scheduler and a probabilistic modeling technique. Extensive experiments across multimodal benchmarks demonstrate the effectiveness of our framework, producing latency- and content-aware execution plans and achieving a range of accuracy-latency tradeoffs. Moreover, our method is compatible with existing efficiency techniques, such as token pruning, further enhancing its practical utility. We believe this work represents a step toward making MLLMs more viable for real-world applications where computational resources may fluctuate significantly.

References

[1] Meta AI. Introducing meta llama 3: The most capable openly available llm to date, 2024. https://ai.meta.com/blog/meta-llama-3/. 4

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022. 1, 2

[3] Anthropic. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024. 1, 5

[4] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015. 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020. 5

[6] Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In *Forty-first International Conference on Machine Learning*, 2024. 1

[7] Jianjian Cao, Peng Ye, Shengze Li, Chong Yu, Yansong Tang, Jiwen Lu, and Tao Chen. Madtp: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15710–15719, 2024. 2

[8] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2025. 1

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 5

[10] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022. 5

[11] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022. 2

[12] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 1039–1048, 2017. 2

[13] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023. 5

[14] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017. 5

[15] Alex Grubb and Drew Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *Artificial Intelligence and Statistics*, pages 458–466. PMLR, 2012. 2

[16] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021. 2

[17] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 5

[18] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 4

[19] Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3812–3821, 2019. 2

[20] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, Singapore, 2023. Association for Computational Linguistics. 5

[21] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 5

[22] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 2023. 2

[23] Zequn Jie, Peng Sun, Xin Li, Jiashi Feng, and Wei Liu. Anytime recognition with routing convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1875–1886, 2019. 2

[24] Sergey Karayev, Mario Fritz, and Trevor Darrell. Anytime recognition of objects and scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 572–579, 2014. 2

[25] Samir Khaki and Konstantinos N Plataniotis. The need for speed: Pruning transformers with one recipe. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[26] Siddique Latif, Moazzam Shoukat, Fahad Shamshad, Muhammad Usama, Yi Ren, Heriberto Cuayáhuitl, Wenwu Wang, Xulong Zhang, Roberto Togneri, Erik Cambria, et al. Sparks of large audio models: A survey and outlook. *arXiv preprint arXiv:2308.12792*, 2023. 2

[27] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. 1, 2

[28] Hengduo Li, Zuxuan Wu, Abhinav Shrivastava, and Larry S Davis. 2d or not 2d? adaptive 3d convolution selection for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6155–6164, 2021. 2

[29] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 1, 2

[30] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023. 5

[31] Yingyu Liang, Heshan Liu, Zhenmei Shi, Zhao Song, Zhuoyan Xu, and Junze Yin. Conv-basis: A new paradigm for efficient attention inference and gradient computation in transformers. *arXiv preprint arXiv:2405.05219*, 2024. 5

[32] Bin Lin, Zhenyu Tang, Yang Ye, Jiaxi Cui, Bin Zhu, Peng Jin, Jinfa Huang, Junwu Zhang, Yatian Pang, Munan Ning, et al. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*, 2024. 1, 2

[33] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023. 1, 2, 5

[34] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 1, 2

[35] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge. `https://llava-vl.github.io/blog/2024-01-30-llava-next/`, 2024. 1

[36] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 5

[37] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, pages 216–233. Springer, 2025. 5

[38] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and

Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022. 5

[39] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. 5

[40] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12309–12318, 2022. 2

[41] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 86–104. Springer, 2020. 2

[42] OpenAI. Introducing ChatGPT. https://openai.com/blog/chatgpt, 2023. Accessed: 2023-09-10. 4

[43] OpenAI. GPT-4 technical report. *arXiv preprint arxiv:2303.08774*, 2023. 1, 4

[44] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022. 5

[45] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red2: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34:24898–24911, 2021. 2

[46] Phu Pham, Wentian Zhao, Kun Wan, Yu-Jhe Li, Zeliang Zhang, Daniel Miranda, Ajinkya Kale, and Chenliang Xu. Quadratic is not what you need for multimodal large language models. *arXiv preprint arXiv:2410.06169*, 2024. 2

[47] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, 2021. 2

[48] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. 2

[49] Daniel Rotem, Michael Hassid, Jonathan Mamou, and Roy Schwartz. Finding the sweet spot: Analysis and improvement of adaptive inference in low resource settings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14836–14851, 2023. 2

[50] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024. 1, 2, 3, 7

[51] Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. Why larger language models do in-context learning differently? In *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023. 5

[52] Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. Why larger language models do in-context learning differently? In *Forty-first International Conference on Machine Learning*, 2024. 5

[53] Fangxun Shu, Yue Liao, Le Zhuo, Chenning Xu, Guanghao Zhang, Haonan Shi, Long Chen, Tao Zhong, Wanggui He, Siming Fu, et al. Llava-mod: Making llava tiny via moe knowledge distillation. *arXiv preprint arXiv:2408.15881*, 2024. 2

[54] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 5

[55] Jiajun Song, Zhuoyan Xu, and Yiqiao Zhong. Out-of-distribution generalization via composition: a lens through induction heads in transformers. *arXiv preprint arXiv:2408.09503*, 2024. 5

[56] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 590–606, 2024. 1

[57] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 4

[58] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 1

[59] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 409–424, 2018. 2

[60] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *Advances in Neural Information Processing Systems*, 2021. 2

[61] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. In *The Eleventh International Conference on Learning Representations*, 2023. 5

[62] Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, En Yu, Jianjian Sun, Chunrui Han, and Xiangyu Zhang.

Small language model meets with reinforced vision vocabulary. *arXiv preprint arXiv:2401.12503*, 2024. 2

[63] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826, 2018. 2

[64] Zhixiang Xu, Kilian Q Weinberger, and Olivier Chapelle. The greedy miser: learning under test-time budgets. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1299–1306, 2012. 2

[65] Zhuoyan Xu, Zhenmei Shi, Junyi Wei, Yin Li, and Yingyu Liang. Improving foundation models for few-shot learning via multitask finetuning. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023. 5

[66] Zhuoyan Xu, Khoi Duc Nguyen, Preeti Mukherjee, Somali Chaterji, Yingyu Liang, and Yin Li. Adainf: Adaptive inference for resource-constrained foundation models. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024. 5

[67] Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. 5

[68] Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability. In *First Conference on Language Modeling*, 2024. 5

[69] Zhuoyan Xu, Zhenmei Shi, Junyi Wei, Fangzhou Mu, Yin Li, and Yingyu Liang. Towards few-shot adaptation of foundation models via multitask finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. 5

[70] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024. 1

[71] Zhengqing Yuan, Zhaoxu Li, Weiran Huang, Yanfang Ye, and Lichao Sun. Tinygpt-v: Efficient multimodal large language model via small backbones. *arXiv preprint arXiv:2312.16862*, 2023. 2

[72] Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, Yan Yan, Beidi Chen, Guangyu Sun, and Kurt Keutzer. Llm inference unveiled: Survey and roofline model insights, 2024. 5

[73] Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. Tinyllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289*, 2024. 2

[74] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2

[75] Yichen Zhu, Minjie Zhu, Ning Liu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Llava-ϕ: Efficient multi-modal assistant with small language model. *arXiv preprint arXiv:2401.02330*, 2024. 2

Learning to Inference Adaptively for Multimodal Large Language Models

Supplementary Material

In the supplementary material, we (1) provide the full set of results accompanying our experiments in Sec. 4 (see Sec. A); (2) provide additional quantitative results on latency-aware Scheduling on MME dataset, and qualitative results on different images (see Sec. B); (3) provide additional attention map results on content awareness (see Sec. C); and (4) provide further discussion of our work (see Sec. D). We hope that this document will complement our main paper.

A. Full Results

We report the full set of results on LLaVA 1.5, LLaVA-PruMerge and LLaVA-PruMerge+ in Tab. 3, as a complementary to Tab. 1. All experiments follows the same setting described in Sec. 4.1. These results confirm that our AdaLLaVA framework successfully adapts to LLaVA 1.5 across different backbone sizes, and can be further combined with recent token selection methods (PruMerge and PruMerge+) to further enhance efficieny. We maintain comparable performance while improving efficiency across multiple benchmarks. Additionally, our analysis reveals how performance varies under different latency constraints, demonstrating our framework's ability to trade between accuracy and latency.

B. Additional Results on Latency-Awareness

We perform further evaluation of AdaLLaVA under varying latency constraints.

MME benchmark. We conduct extensive experiments on MME benchmark to demonstrate the adaptive ability of our approach, complementary to Sec. 4.3.

We mainly adopt LLaVA 1.5-7B with and without PruMerge to show the latency awareness of our AdaLLaVA framework, while compatible with token selection techniques, effectively balances accuracy and efficiency across different latency requirements.

For LLaVA 1.5 base model and PruMerge, the upward trend in the AdaLLaVA-L results confirms our scheduler's effective component selection: as computational resources increase, it activates additional components to enhance performance. Notably, our AdaLLaVA-L model surpasses the base model's performance even at lower latency (around 1.1T FLOPs). This demonstrates our framework's content awareness, generating customized execution plans for different inputs under the same latency constraints - ultimately outperforming the fixed full model. In contrast, the AdaLLaVA-H variant shows less predictable patterns across different latency budgets. We attribute this to the relative simplicity of the MME benchmark compared to VQAv2 - the model can achieve satisfactory performance even with limited computational resources. Overall, AdaLLaVA-H offers more flexibility in terms of latency requirements.

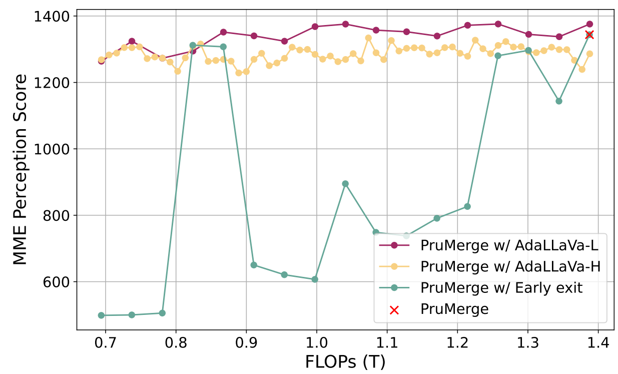


Figure 7. Results on LLaVA 1.5-7b.

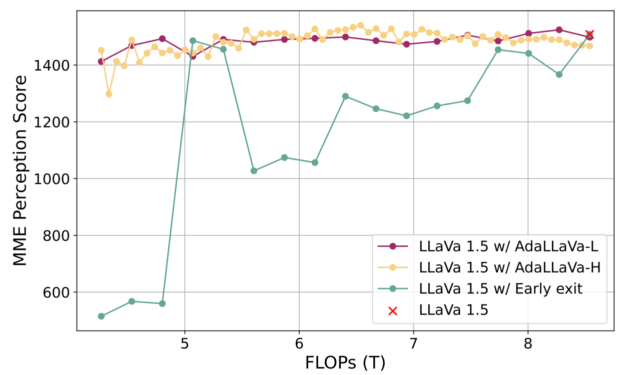


Figure 8. Results on LLaVA 1.5-7b with PruMerge.

Figure 9. Results on MME benchmark across latency budgets (FLOPs). AdaLLaVA-L/H: our methods. We denote performance of full model as a single point.

Model Response under different latency. Here we show additional results on model response given same image-text input under different latency budget, similar to Fig. 1.

As shown in Tab. 4, given an image-query pair and latency constraints, AdaLLaVA learns to generate appropriate responses while adapting to varying computational budgets.

Table 3. Full results of AdaLLaVA across six benchmarks.

Method	LLM	Percentage (%)	VQA^{v2}	SQA^{I}	VQA^{T}	POPE	MME	MMB
BLIP-2	Vicuna-13B	-	41.0	61	42.5	85.3	1293.8	-
InstructBLIP	Vicuna-7B	-	-	60.5	50.1	-	-	36
InstructBLIP	Vicuna-13B	-	-	63.1	50.7	78.9	1212.8	-
Shikra	Vicuna-13B	-	77.4	-	-	-	-	58.8
IDEFICS-9B	LLaMA-7B	-	50.9	-	25.9	-	-	48.2
IDEFICS-80B	LLaMA-65B	-	60.0	-	30.9	-	-	54.5
Qwen-VL	Qwen-7B	-	78.8	67.1	63.8	-	-	38.2
Qwen-VL-Chat	Qwen-7B	-	78.2	68.2	61.5	-	1487.5	60.6
LLaVA-1.5	Vicuna-7B	100	78.5	66.8	58.2	85.9	1510.7	64.3
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-7B	100	78.4	67.7	56.6	86.5	1498.2	64.3
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-7B	85	77.3	68.1	53.9	86.4	1505.3	64.3
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-7B	75	76.0	67.4	51.1	85.6	1498.9	63.7
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-7B	60	73.5	66.6	45.4	85.2	1490.2	61.9
LLaVA-1.5 w/ `AdaLLaVA-H`	Vicuna-7B	100	77.3	68.3	55.5	85.8	1467.5	63.1
LLaVA-1.5 w/ `AdaLLaVA-H`	Vicuna-7B	85	76.9	67.9	54.5	86.0	1502.6	62.5
LLaVA-1.5 w/ `AdaLLaVA-H`	Vicuna-7B	75	75.0	67.5	50.6	86.0	1524.9	62.4
LLaVA-1.5 w/ `AdaLLaVA-H`	Vicuna-7B	60	72.1	67.0	45.6	86.6	1480.7	61.8
Prumerge	Vicuna-7B	100	72.0	68.5	56.0	76.3	1350.3	60.9
Prumerge w/ `AdaLLaVA-L`	Vicuna-7B	100	69.4	68.3	53.1	74.4	1375.7	56.6
Prumerge w/ `AdaLLaVA-L`	Vicuna-7B	85	68.6	68.6	51.8	74.0	1375.7	57.6
Prumerge w/ `AdaLLaVA-L`	Vicuna-7B	75	67.4	67.8	48.8	73.3	1375.6	56.3
Prumerge w/ `AdaLLaVA-L`	Vicuna-7B	60	65.6	68.4	44.1	75.6	1351.5	55.6
Prumerge w/ `AdaLLaVA-H`	Vicuna-7B	100	68.3	68.6	52.8	70.5	1286.3	57.4
Prumerge w/ `AdaLLaVA-H`	Vicuna-7B	85	67.7	68.2	51.3	69.2	1278.8	57.0
Prumerge w/ `AdaLLaVA-H`	Vicuna-7B	75	66.0	68.4	48.7	70.5	1269.2	56.0
Prumerge w/ `AdaLLaVA-H`	Vicuna-7B	60	63.7	68.0	44.3	72.3	1269.4	54.7
Prumerge+	Vicuna-7B	100	76.8	68.3	57.1	84.0	1462.4	64.9
Prumerge+ w/ `AdaLLaVA-L`	Vicuna-7B	100	75.1	68.7	53.9	82.4	1455.8	62.0
Prumerge+ w/ `AdaLLaVA-L`	Vicuna-7B	85	74.5	67.9	53.0	82.5	1472.3	61.6
Prumerge+ w/ `AdaLLaVA-L`	Vicuna-7B	75	73.3	67.9	50.4	80.8	1441.7	61.9
Prumerge+ w/ `AdaLLaVA-L`	Vicuna-7B	60	71.4	67.5	46.3	83.3	1459.2	61.8
LLaVA-1.5	Vicuna-13B	100	80.0	71.6	61.3	85.9	1531.3	67.7
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-13B	100	79.4	72.3	58.9	86.3	1497.9	68.7
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-13B	85	79.1	72.6	58.1	86.1	1519.3	68.3
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-13B	75	78.4	72.5	56.6	86.2	1506.3	69.0
LLaVA-1.5 w/ `AdaLLaVA-L`	Vicuna-13B	60	77.1	71.9	54.7	86.9	1517.1	68.5
Prumerge	Vicuna-13B	100	72.8	71.0	58.4	78.5	1428.2	62.3
Prumerge w/ `AdaLLaVA-L`	Vicuna-13B	100	70.8	72.4	55.4	72.3	1343.5	60.7
Prumerge w/ `AdaLLaVA-L`	Vicuna-13B	85	70.4	72.7	55.3	71.5	1347.0	61.1
Prumerge w/ `AdaLLaVA-L`	Vicuna-13B	75	69.9	72.5	54.1	72.5	1355.0	60.7
Prumerge w/ `AdaLLaVA-L`	Vicuna-13B	60	68.9	72.7	51.6	72.5	1350.7	60.0

C. Additional Results on Content-Awareness

We provide additional results on content awareness by showing the key-query attention scores of the latency token and the input visual tokens with different text questions, similar to Fig. 4

Fig. 10 further demonstrate the model's content-aware behavior. For in the father-child scene, attention spreads across the entire street view for scene description but concentrates on the middle when asking about their activity. For the Happy Plaza image, attention focuses on the storefront sign when asking about the location name, but shifts to the promotional signage area when querying about special offers. Similarly, in the restaurant scene, attention distributes across the interior elements (tables, counter, chairs) when identifying the location type, but concentrates specifically on the woman's clothing when asked about her attire. This consistently shows how the model adjusts its attention based on query requirements.

D. Further Discussion

Modeling the Latency Constraints: Deterministic vs. Probabilistic. We further elaborate different approaches for

Visual input example:			
User	 Describe this image.	 Describe this image.	 Describe this image.
	The image features an older man sitting at a table, surrounded by various items. He is using a cell phone, possibly checking messages or browsing the internet...	The image features a wooden block with a set of wooden letters arranged in a way that spells out the word "press stop." ...	The image features a woman walking down a street, holding a bouquet of flowers in her hand...
AdaLLaVA-L-50%	The image depicts a man sitting at a table with a large number of papers and books on it. He is using a cell phone, possibly checking or communicating with someone. The man is wearing a tie, which adds to the professional setting. The table is surrounded by several chairs, with one close to the man and others further away. A clock can be seen on the left side of the table, and a handbag is placed on the right side. The scene appears to be a workspace or a study area, with the man working on his tasks and using the cell phone.	The image features a wooden board with a series of wooden chicks on it. The wooden chicks are placed in a row, with each one occupying a different position on the board. The arrangement of the chicks is a close-up of the word "press" and the word "stap.	The image captures a young woman standing in a busy street, holding a bouquet of flowers. She appears to be smiling, possibly indicating that she is enjoying the moment. The street is filled with various vehicles, including several motorcycles and a motorcycle parked close to the woman. There are numerous people in the scene, some standing closer to the woman, while others are further away. Some of the people are closer to the left side of the image, while others are further back, creating a sense of a busy city street.
AdaLLaVA-L-75%	The image features a man sitting at a table, surrounded by various items. He is wearing a suit and tie, and appears to be using a cell phone. The table is set with a clock, a cup, and a book. There are also two chairs in the scene, one near the man and another further away. In the background, there are two more books placed on the ground, and a handbag is located near the edge of the scene. The man seems to be engaged in a task or a conversation, as evidenced by his use of the cell phone.	The image features a wooden word puzzle made up of four wooden letters, each with a number on them. The letters are arranged in a way that reads "Press Stop." The letters are placed on a white background, creating a visually appealing and playful display. The arrangement of the letters and numbers adds a unique and creative touch to the word puzzle.	The image depicts a woman walking down a street, holding a bouquet of flowers in her hand. She is wearing a dress and appears to be enjoying her time outdoors. The street is lined with various vehicles, including motorcycles and a truck, parked along the side. There are several people in the scene, some of whom are walking or standing near the vehicles. A handbag can be seen resting on the ground, possibly belonging to one of the individuals. The overall atmosphere of the scene is lively and bustling, with people going about their daily activities.
AdaLLaVA-L-100%	The image features an older man sitting at a table, working on a laptop. He is surrounded by various items, including a cup, a cell phone, and a book. The table is adorned with a red and black tablecloth, and there are several signs and posters on the wall behind him. In the room, there are two chairs, one near the man and another further away. A clock is also visible on the wall, indicating the time. The scene appears to be a workspace or a study area, with the man focused on his tasks.	The image features a wooden tray with a set of wooden letters arranged on it. The letters spell out the word "Press Stop," with each letter positioned next to the other. The arrangement of the letters creates a visually appealing and organized display. The wooden tray itself is placed on a white surface, which further emphasizes the contrast between the wooden letters and the background.	The image depicts a woman walking down a street, holding a bouquet of flowers in her hand. She is wearing a flowered dress and appears to be enjoying her time. The street is lined with various vehicles, including several motorcycles parked on both sides of the road. There are also a few people walking around, some of whom are carrying handbags. The scene captures a lively atmosphere with people going about their daily activities.

Table 4. AdaLLaVA-L on LLaVA 1.5-7b model, generating appropriate responses while adapting to varying computational budgets.

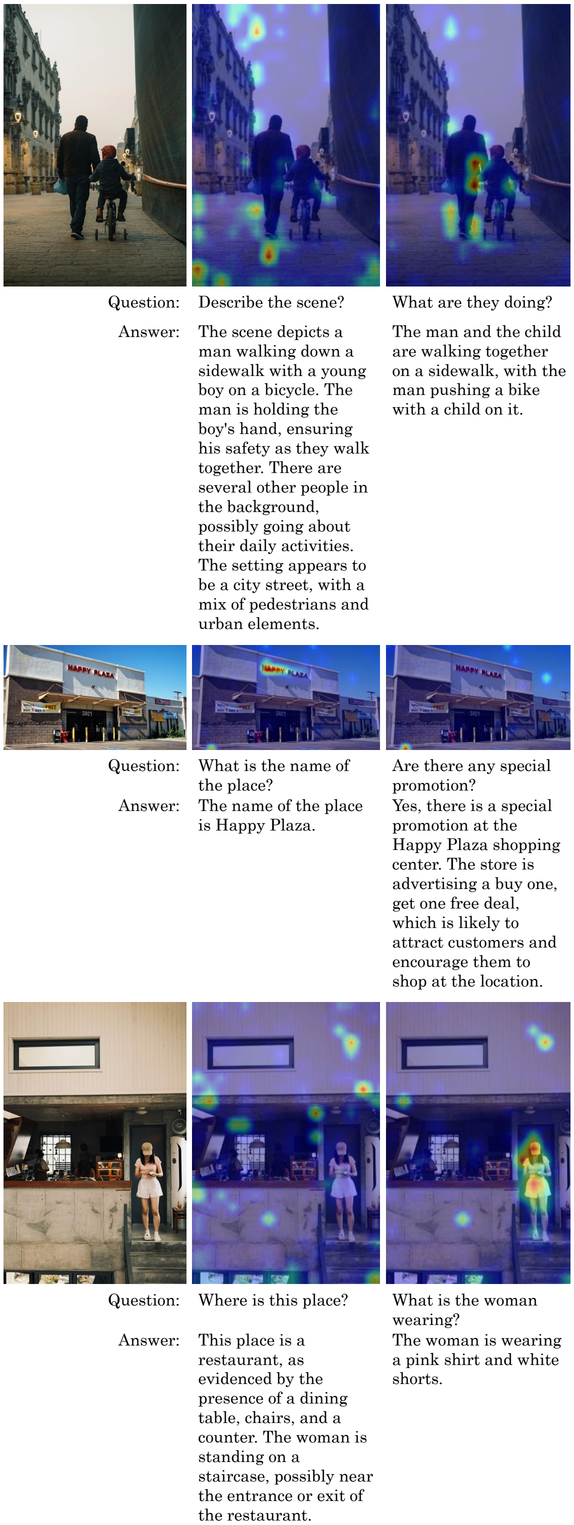


Figure 10. The key-query attention scores between latency token and visual tokens. The latency input is 1.0 in these examples.

modeling latency constraint when learning the scheduler. In Sec. 3.3, we have described a relaxed version of the constrained optimization using a deterministic, optimization-based approach that minimizes the following loss function

$$\arg\min_{\theta,\phi} \; -\Sigma_i \log p\left(x_i^a = f(\cdot)\right) + \lambda \|\text{Latency}(f(\cdot)) - l\|_2^2.$$

In addition to this loss, we have also experimented with a different training objective using a hinge loss to encode the latency constraints.

$$\arg\min_{\theta,\phi} \; -\Sigma_i \log p\left(x_i^a = f(\cdot)\right) + \lambda max(0, \text{Latency}(f(\cdot)) - l).$$

Both approaches lead to similar results. The learned schedulers often fail to adhere to the latency budget, or do not fully use the budget and thus show worse accuracy. We conjecture that the main issue is that the relaxed constrained optimization can not handle hard latency constraint, and may overlook the balance between accuracy and latency.

In contrast, our probabilistic approach demonstrates superior adaptability across different latency budgets compared to theses deterministic approach. This is clearly shown in our ablation results in Tab. 2, Sec. 4.4. It is perhaps interesting to note that our sampling process that enforces the latency constraint is conceptually similar to a projection step in projected gradient descent, in which a feasible solution satisfying the constraint is always produced given a initial solution. This ensures both constraint compliance and improved performance under varying conditions.

Limitation and Future Directions. With 100% latency, our approach can outperform the base model on some datasets (*e.g.* MME), yet may fall slightly behind the base model's performance on others (*e.g.* VQAv2), as shown in Figs. 3a and 3b. This is likely because the full model has already reached optimal performance on VQAv2. The adaptation mechanism, while successful in creating efficiency gains, introduces minor perturbations from the optimal solution in the optimization landscape. We will further investigate this direction.

Another promising direction is to explore LoRA [18]-based fine-tuning for our approach, instead of full fine-tuning as currently considered in our experiments. Doing so will lead to adapters that are separated from the base MLLM, *i.e.*, the pre-trained parameters and the architecture of the base MLLMs remain unchanged. If successful, this will allow us to design multiple adapters for the same MLLMs, with each potentially tailored for one deployment scenario (*e.g.*, server farm vs. edge device).

E. Additional Related Work

Large Language Models. Large Language Models (LLMs) are typically based on the Transformer architecture and are characterized by their enormous number of parameters and extensive pretraining on vast datasets. Notable examples include LLaMA[1, 57], ChatGPT [42], GPT4 [43] and Claude [3]. These models utilize various pretraining methods such as masked language modeling [9, 36], and autoregressive pretraining [5]. Researchers have investigated the effects of pretraining on language model performance. Adapting LLMs to various downstream tasks has garnered significant attention in the field. This adaptation can take many forms, including the use of adapters [17, 20], multitask fine-tuning [61, 65, 69], in-context learning [10, 51, 52, 55, 67, 68], reinforcement learning from human feedback (RLHF) [44], and methods for accelerating inference [31, 66]. Each of these approaches aims to enhance LLM performance or efficiency for specific applications or domains, allowing these powerful models to be tailored to a wide range of tasks and requirements.