Learning to Inference Adaptively for Multimodal Large Language Models

Zhuoyan Xu[1]*, Khoi Duc Nguyen[1]*, Preeti Mukherjee[2],
Saurabh Bagchi[2], Somali Chaterji[2], Yingyu Liang[1,3], Yin Li[1]
[1]University of Wisconsin-Madison [2]Purdue University [3]The University of Hong Kong

Abstract

*Multimodal Large Language Models (MLLMs) have shown impressive capabilities in visual reasoning, yet come with substantial computational cost, limiting their deployment in resource-constrained settings. Despite recent effort on improving the efficiency of MLLMs, prior solutions 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 conduct extensive experiments across benchmarks involving question-answering, reasoning, and hallucination. Our results show that AdaLLaVA effectively adheres to input latency budget, achieving varying accuracy and latency tradeoffs at runtime. Further, we demonstrate that AdaLLaVA adapts to both input latency and content, can be integrated with token selection for enhanced efficiency, and generalizes across MLLMs. Our project webpage with code release is at https://zhuoyan-xu.github.io/ada-llava/.*

1. Introduction

Large language models (LLMs) [2, 43] have recently been extended to connect visual and textual data, giving rise to multimodal large language models (MLLMs). Exemplified by LLaVA [35, 36] and other works [1, 30, 32, 37, 54, 69], MLLMs have shown impressive visual reasoning capabilities, but come with significant computational costs. Several efforts have sought to improve the efficiency of MLLMs by exploring lightweight architectures, mixture of experts, or token selection techniques [8, 34, 49, 63, 69]. However, prior approaches typically exhibit a fixed accuracy and latency footprint during inference, rendering them incapable of adapting to varying compute budget or input content.

We argue that MLLMs with fixed computational footprints are insufficient for real-world deployment. Consider an example of deploying an MLLM on a server farm. Different requests may have different latency requirements, *e.g.*, requests from a mobile application require instant feedback to users, while asynchronous processing tasks such as video summarization can tolerate higher latency due to their non-interactive nature. Further, the available computing resources may vary over time as the overall load on the system fluctuates. Similarly, when deployed on an edge

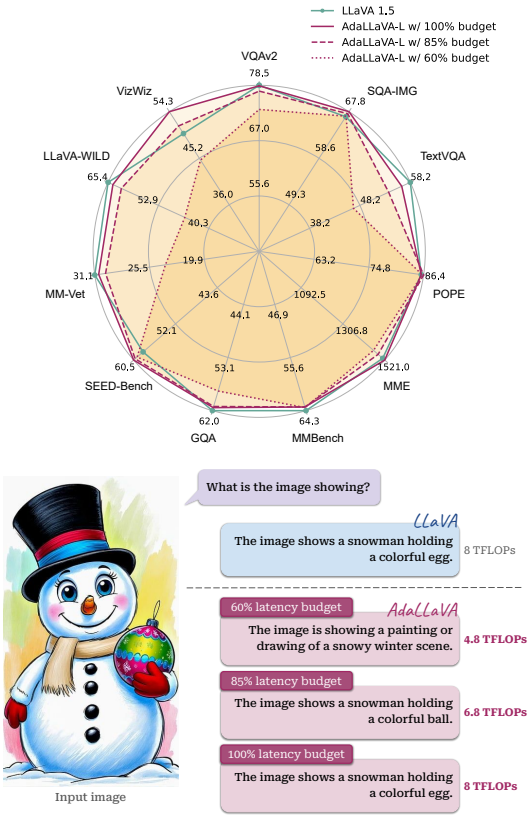

Figure 1. **Top**: **AdaLLaVA** empowers a base LLaVA model with the ability to adapt to varying compute budgets at inference time while maintaining minimal performance loss. **Bottom**: Given an image, a text query and a latency budget, AdaLLaVA learns to reconfigure operations within a base MLLM, generating appropriate responses while sticking to the budget.

*Equal contribution

device, the latency budget often remains constant, yet the computing resources may vary due to contention produced by other concurrent programs. In spite of this need, developing adaptive inference strategies for MLLMs that are robust across varying computational budgets [25] remains an open research challenge.

To bridge this gap, we propose *latency-aware adaptive inference for MLLMs*, aiming to dynamically adjust a model's computational load based on input content and a specified latency or compute budget.[1] This problem is of both conceptual interest and practical significance. Our key insight is that a modern MLLM can be viewed as a collection of shallower models, where choosing among these models enables 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 minimal impact on accuracy, while reducing latency [5, 26, 52]. Therefore, strategically selecting these operations during inference results in a set of models with shared parameters but distinct accuracy-latency tradeoffs, allowing the MLLM to flexibly respond to varying latency budgets and content complexity.

To this end, we present **AdaLLaVA**, a learning-based framework for adaptive inference in MLLMs. As shown in Fig. 1, given an input image, a text query, and a latency budget, AdaLLaVA empowers an MLLM to answer the query about the image while adhering to the specified budget — a capability unattainable with the base MLLM. The 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 budget while maximizing expected accuracy. To enable effective learning of the scheduler, we introduce a probabilistic formulation in tandem with a dedicated sampling strategy, designed to account for latency constraints at training time.

We conduct extensive experiments to evaluate AdaLLaVA. Our results demonstrate that AdaLLaVA can achieve a range of accuracy-latency tradeoffs at runtime. AdaLLaVA exhibits strong adaptability to different latency budgets, effectively trading accuracy for compute during inference. Across several benchmarks, AdaLLaVA retains comparable performance to its base MLLM while operating with higher efficiency (see Fig. 1). For example, on several comprehensive benchmarks, AdaLLaVA can achieve 99.0% and 98.2% average performance of the baseline LLaVA model when using only 80% and 65% of the latency budget, respectively.

Importantly, it consistently adheres to specified latency constraints and generates content-aware execution plans tailored to input images. Furthermore, we show that AdaLLaVA can be integrated with existing token selection techniques designed to enhance efficiency, making it a versatile solution for adaptive inference in MLLMs.

Our key **contributions** are summarized as follows.

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

2. Related Work

Multimodal large language models (MLLMs). There has been a growing interest in extending text LLMs to multimodal signals, including images [35], video [30], and audio [27]. This leads to the emergence of MLLMs, often involving combining vision encoders with existing LLMs. Flamingo [1] inserts gated cross-attention dense blocks between vision encoder and LLMs to align vision and language modality. BLIP2 [32] introduces Q-former with two-stage pretraining, bridging frozen image encoders and LLMs to enable visual instruction capability. LLaVA [35, 36] and MiniGPT-4 [69] use a simple MLP to connect vision embedding and text token, achieving impressive performance across various tasks. Our work builds on these developments and aims to enable adaptive inference in MLLMs under varying latency budgets.

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 level [18]. Early works focused on the selection of hand-crafted features in multi-stage prediction pipelines [15, 25, 62]. More recent works have extended these ideas to deep models. For convolutional networks, methods have been developed to selectively downsample inputs, skip layers, or exit early during inference [4, 12, 19, 24, 31, 42, 55, 59–61]. For vision Transformers, various approaches have been proposed to select different image patches [44, 46, 56], or choose different attention heads and blocks [26, 41]. Similar ideas have also been explored for LLMs and recently MLLMs, where models selectively process tokens [47, 67] or execute a subset of

[1] In this paper, we measure a model's latency and its budget using the number of floating-point operations (FLOPs). Thus, the terms "compute budget" and "latency budget" are used interchangeably throughout.

the operations [11, 48] during inference.

Our approach is conceptually similar to existing methods by dynamically selecting a subset of model components during inference. Yet unlike prior methods, our work specifically targets the latency-aware inference of MLLMs, predicting feasible execution plans tailored for input while adhering to varying latency budgets.

Efficient inference for MLLMs. MLLMs face a major challenge in deployment due to their high computational costs during inference. Several works have designed lightweight model architectures to reduce the costs. Examples include Phi-2 [23], TinyGPT-V [65] and LLaVA-ϕ [71]. Vary-toy [58] enhances performance through specialized vision vocabulary in smaller models. TinyLLaVA [68] and LLaVA-OneVision [30] learn small-scale models with curated training data and pipeline. MoE-LLaVA [34] and LLaVA-MoD [50] improve efficiency by incorporating mixture-of-experts architectures and parameter sparsity techniques. Recent works also investigate input token selection, as an input image or video can produce a large number of vision tokens. MADTP [6] and LLaVA-PruMerge [49] introduce token pruning and merging technique to reduce the tokens counts. Recently, Pham et al. [45] propose to selectively disabling attention mechanisms for visual tokens in MLLMs.

While our approach also aims to improve the efficiency of MLLMs, it focuses on dynamically adjusting an MLLM to fit varying latency budgets during inference. This makes our approach orthogonal to prior efforts on developing inherently more efficient MLLMs. Through our experiments, we will demonstrate that our approach is compatible with lightweight models and integrates seamlessly with existing token-pruning techniques (*e.g.*, [8, 49]).

3. Adaptive Inference of MLLMs

We now present **AdaLLaVA**, our adaptive inference framework for MLLMs. Given a latency budget and an input image-query pair at inference time, AdaLLaVA leverages a scheduler learned from data to dynamically reconfigure the execution of MLLMs. Importantly, this scheduler strategically selects a subset of operations to execute, catered to the input budget and content. In doing so, AdaLLaVA 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 an input of both multimodal sample and latency budget, and outputs an execution plan. 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 details of our solution (Sec. 3.5).

3.1. Preliminaries: MLLMs

An 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 $h_v(\cdot)$ (including a vision backbone and its projector) 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 the words x^q into a set of text tokens $\{\mathbf{z}^q \in \mathbb{R}^d\}$ with $\mathbf{z}^q = h_t(x^q)$. These tokens are combined into $\{\mathbf{z}^{v|q}\} = [\{\mathbf{z}^v\}, \{\mathbf{z}^q\}]$, and processed by an LLM $f(\cdot)$, which decodes the answer $\mathbf{X}^a$ 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 of our 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 an MLLM can be conceptualized as a collection of shallower models with shared parameters, each offering a distinct accuracy-latency tradeoff. This perspective enables dynamic reconfiguration of the MLLM during inference to meet varying latency budgets. To this end, we propose equipping the LLM $f(\cdot)$ with K tunable binary switches $\mathbf{s} \in (0, 1)^K$, which control the execution of individual operations, such as Transformer blocks or attention heads, at runtime. Each switch determines whether a specific operation will be executed (1) or skipped (0). We defer the choice of these operations and the design of these switches to our model instantiation. Here, we focus on the concept of reconfigurable LLM decoding, expressed as

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

Specifically, $f(\cdot)$ now takes the switches $\mathbf{s}$ as an additional input, and selectively executes a subset of operations when generating its output. Note that the switches $\mathbf{s}$ do not depend on the decoding step i, *i.e.*, given the input tokens, a fixed set of operations is applied to generate all output tokens, although the operations may vary for different inputs.

Scheduler. The core of our method is a scheduler $g(\cdot)$ that controls the execution of $f(\cdot)$ during inference. The scheduler $g(\cdot)$ is trained to predict a configuration of switches $\mathbf{s}$ based on the 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 the parameters of the scheduler $g(\cdot)$.

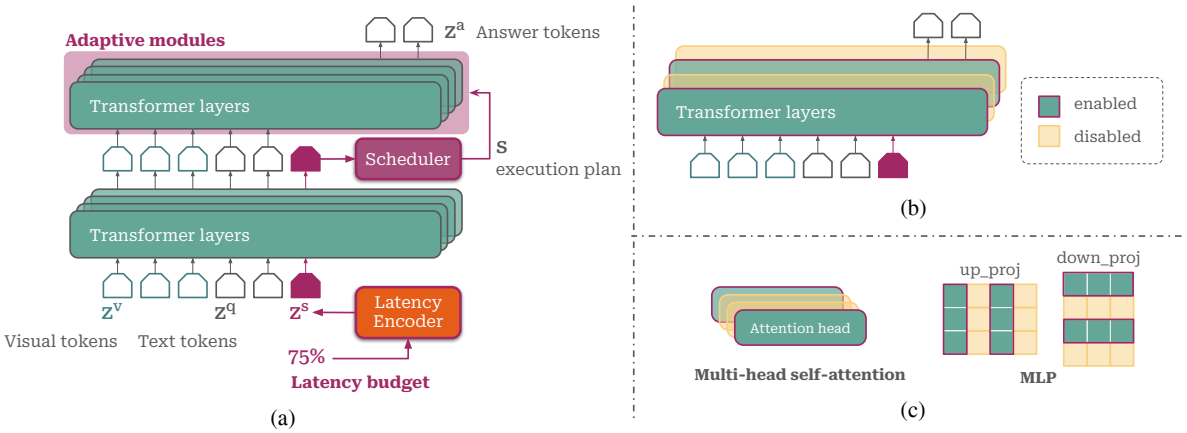

Figure 2. Overview of **AdaLLaVA**. **(a) Model architecture**: Our latency encoder embeds an input latency budget into a latency token, which is further processed by the early part of the LLM. The resulting embedding is then fed into the scheduler, leading to the output of an execution plan that controls individual operations in the remaining part of the LLM. Our latency encoder and scheduler are jointly trained with the MLLM. **(b) `AdaLLaVA-L`**: the scheduler controls the execution of entire Transformer blocks. **(c) `AdaLLaVA-H`**: the scheduler controls the execution of attention heads and MLP neurons, by masking out their activation values and the corresponding weights.

The goal of $g(\cdot)$ is to determine an execution plan that meets the latency requirement while maximizing the accuracy. This requires solving the following combinatorial optimization problem *for each input sample*:

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

The objective here is to minimize the negative log likelihood of the target token—the standard loss used for training MLLMs, while the constraint ensures that the latency of executing the model falls within the budget.

3.3. Learning to Schedule Execution Plans

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

Deterministic modeling. One possible solution is to solve a relaxed version of the constrained optimization at training time. We initially explored this solution, where we task $g(\cdot)$ to predict a hard execution plan with binary switches $\mathbf{s}$ and attribute latency violation as part of the objective. This leads to the following loss

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

where λ can be treated as the Lagrange multiplier. The execution of the LLM $f(\cdot)$ relies on the output from the scheduler $g(\cdot)$, allowing the joint optimization of $f(\cdot)$ and $g(\cdot)$.

We empirically found that this method fails to enforce a strict latency constraint on the scheduler and often produces suboptimal execution plans that under-utilize the available resources. We demonstrate this limitation through experimental results in Sec. 4.3.

Probabilistic modeling. In contrast, we propose a probabilistic model to further relax the constraints, avoiding directly solving Eq. (4) while stabilizing the joint training of the LLM and the scheduler. Specifically, we task $g(\cdot)$ to model a distribution over the choice of the switches $\mathbf{s}$, in lieu of making a hard decision:

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

With slight abuse of notation, we denote $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)$ as the probability over the states $\mathbf{s}$ of K binary switches 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 of $\mathbf{s}$ exceeds the budget l.

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

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

Conceptually, this formulation defines the following generative process: (1) the scheduler g considers the input and a latency budget and outputs the conditional probability of the execution plan $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)$; (2) an execution plan $\mathbf{s}$ is then sampled from the predicted distribution without violating the latency constraint; and (3) the plan is executed to sequentially decode x_i^a and generate the answer.

Modeling $p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right)$. Our design requires that the sampled execution plan strictly adheres to the latency budget while maximizing resource utilization. To achieve this,

we restrict the support of $p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right)$ to the states s that have exactly k activated switches, where k is the maximum number of switches allowed to be turned on without violating l. Specifically, to sample $\mathbf{s}$, $g\left(\{\mathbf{z}^{v|q}\}, l; \phi\right)$ first outputs a categorical distribution over K available switches. Then, k switches are picked one by one without replacement, following the categorical distribution.

Training loss. The probabilistic model allows us to directly train the LLM and the scheduler with the following loss

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

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

$$\begin{aligned} &p(x_i^a | [\{\mathbf{z}^{v|q}\}, \{\mathbf{z}_{<i}^a\}], l, \theta, \phi)) = \\ &\quad \mathbb{E}_{p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)} \left[p(x_i^a | [\{\mathbf{z}^{v|q}\}, \{\mathbf{z}_{<i}^a\}], \mathbf{s}, \theta)\right]. \end{aligned} \tag{7}$$

Thus, the loss function is transformed into

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

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

3.4. Training and Inference

Approximate training. We present an approximate training scheme in the context of stochastic gradient descent (SGD). Specifically, for each training sample within a mini-batch, a latency budget l is first sampled uniformly from a range of possible budgets, then an execution plan $\mathbf{s}$ is sampled from $p\left(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi\right)$. With the sampled $\mathbf{s}$ guaranteed to satisfy the budget l, the next token x_i^a can be decoded and the log-likelihood $\log p\left(x_i^a | \left[\{\mathbf{z}^{v|q}\}, \{\mathbf{z}_{<i}^a\}\right], \mathbf{s}, \theta\right)$ (*i.e.*, the loss) can be readily computed. Optimizing this loss requires backpropagation 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 [22, 40]. See the supplement for more details.

Adaptive inference. During inference, the scheduler outputs the probability $p(\mathbf{s}|\{\mathbf{z}^{v|q}\}, l, \phi)$ over possible switch configurations $\mathbf{s}$, given the input $\{\mathbf{z}^{v|q}\}$ and the latency budget l. In theory, decoding the answer $\mathbf{X}^a$ requires marginalizing over this distribution, which is infeasible due to the large number of configurations. In practice, we approximate the inference by selecting the most probable execution plan from the scheduler. This approximation bypasses the marginalization and thus remains highly efficient. We empirically verify its effectiveness. Formally, this approximation is given by

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

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

3.5. Model Instantiation

Design of tunable switches. We consider attaching binary switches to the LLM part of an MLLM, which accounts for the majority of computational costs. We explore two different designs of switches to select operations.

- `AdaLLaVA-L` (layer-level): This design attaches binary switches 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 thus determines whether each layer is computed or bypassed (see Fig. 2(b)).
- `AdaLLaVA-H` (head/neuron-level): This design introduces binary switches within Transformer blocks, targeting individual attention heads in attention modules and specific neurons in MLP layers. When a switch is off, its computation is skipped, and its contribution is removed. In MLP, switches function similarly to dropout [53], selectively disabling neuron activations (see Fig. 2(c)).

Model architecture. Our goal is to design a lightweight scheduler that minimizes computational overhead yet remains expressive enough to support effective decision-making. To this end, we reuse part of the LLM $f(\cdot)$ to extract visual-language features and encode the latency constraint for the scheduler. Specifically, we first design a latency encoder that converts a latency budget into a token embedding, which is then appended to the original input sequence before being processed by the LLM layers. Within the LLM, the latency token is processed by a few Transformer blocks, attending to all visual-language tokens. The processed token is then passed to a lightweight scheduler that generates the execution plan for the rest of the LLM. Notably, the first few Transformer blocks in the LLM serve two purposes: it simultaneously processes regular MLLM tasks and learns resource allocation based on both content and budget constraints. This design is depicted in Fig. 2 (a).

Implementation details. Our *latency encoder* uses the sinusoidal positional encoding [57] to map the scalar latency l to a 256-D vector. A two-layer MLP, with GELU and layer norm, then converts this vector to a latency token $\mathbf{z}^s$, ready to be appended to the input sequence of the LLM (see Fig. 2(a)). Our *scheduler* is implemented as a linear layer that maps the processed latency token (from the bottom part of the LLM Transformer blocks) to logits, defining a categorical distribution over switch selection. We use *FLOPs* to quantify the theoretical latency budget, following the calculation in [66]. Specifically, we report the average prefill FLOPs on a target dataset, isolating it from variations in decoding length to ensure a more consistent evaluation.

We split the LLM evenly into two parts unless otherwise specified. We use the first part to process the latency token, and apply tunable switches exclusively to the latter part. In AdaLLaVA-H, we attach a switch to each attention head

Method	LLM	Budget (%)	FLOPs (T)	Prefill time (ms)	VQA^{v2} [14]	SQA^{I} [39]	VQA^{T} [51]	POPE [33]	MME [13]	MMBench [38]
LLaVA-1.5 [36]	Vicuna-7B	100	8.6	81	78.5	66.8	58.2	85.9	1510.7	64.3
w/ AdaLLaVA-L	Vicuna-7B	100	8.6	81	78.4	67.8	57.0	85.9	1521.0	63.7
w/ AdaLLaVA-L	Vicuna-7B	85	7.2	69	77.1	67.4	54.5	86.4	1487.2	63.7
w/ AdaLLaVA-L	Vicuna-7B	60	5.1	49	75.0	66.9	47.7	86.1	1463.8	63.8
w/ AdaLLaVA-H	Vicuna-7B	100	8.6	81	77.9	68.5	57.1	86.9	1471.1	64.1
w/ AdaLLaVA-H	Vicuna-7B	85	7.2	69	76.8	68.2	55.2	86.7	1494.9	64.3
w/ AdaLLaVA-H	Vicuna-7B	60	5.1	49	74.2	68.1	48.7	85.0	1489.6	64.8
Prumerge+ [49]	Vicuna-7B	100	3.0	29	76.8	68.3	57.1	84.0	1462.4	64.9
w/ AdaLLaVA-L	Vicuna-7B	100	3.0	29	76.3	68.3	55.8	85.1	1455.5	61.9
w/ AdaLLaVA-L	Vicuna-7B	85	2.6	24	75.3	68.5	52.9	85.7	1429.5	62.5
w/ AdaLLaVA-L	Vicuna-7B	60	1.8	17	73.0	67.7	47.4	85.6	1450.9	61.3
w/ AdaLLaVA-H	Vicuna-7B	100	3.0	29	76.0	67.9	56.0	86.6	1503.2	63.2
w/ AdaLLaVA-H	Vicuna-7B	85	2.6	24	75.0	68.1	54.2	86.4	1511.8	63.6
w/ AdaLLaVA-H	Vicuna-7B	60	1.8	17	72.2	67.6	47.2	86.4	1458.0	63.6
FastV (K=2,R=0.5) [8]	Vicuna-7B	100	4.9	47	77.7	68.7	58.1	82.5	1516.2	64.3
w/ AdaLLaVA-L	Vicuna-7B	100	4.9	47	77.8	67.7	57.0	82.8	1494.3	63.5
w/ AdaLLaVA-L	Vicuna-7B	85	4.2	40	76.9	67.8	54.4	83.3	1478.1	63.7
w/ AdaLLaVA-L	Vicuna-7B	60	3.0	29	74.5	67.0	47.4	83.8	1463.1	63.2
w/ AdaLLaVA-H	Vicuna-7B	100	4.9	47	77.4	68.4	57.0	84.3	1484.2	63.8
w/ AdaLLaVA-H	Vicuna-7B	85	4.2	40	76.6	67.7	54.8	83.9	1520.5	63.9
w/ AdaLLaVA-H	Vicuna-7B	60	3.0	29	73.9	68.3	48.7	82.4	1452.8	65.3

Table 1. **Results on MLLM benchmarks**. Budget (%): input latency budget w.r.t. the base model latency. 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+ and FastV both use LLaVA 1.5. AdaLLaVA enables a base MLLM to adapt to varying latency budgets with competitive performance, and can be further integrated with token selection to enhance overall efficiency.

in the self-attention. For the MLP, channels are grouped to match the number of attention heads, with each group controlled by a single switch This implementation reduces the design space while preserving control granularity. See ablation study on group size in the supplement.

4. Experiments and Results

We now present our experiments and results. We introduce our setup (Sec. 4.1), present our main results (Sec. 4.2), and provide further analyses (Sec. 4.3). Additional experiments, including further ablations, are included in our supplement.

4.1. Experimental Setup

Experiment protocol. In most of our experiments, we build on LLaVA-1.5 [36]. Training LLaVA and many other MLLMs typically involves two stages: (1) vision-language alignment pre-training; and (2) visual instruction tuning. We focus on the second stage and seek to jointly finetune the LLM within the MLLM and train our scheduler using visual instruction data, while keeping the vision encoder frozen. Once trained, we perform zero-shot inference across multiple benchmarks following the common practice in the community [36], yet under varying latency budgets.

Training details. Our model is initialized with the pre-trained LLaVA-1.5 checkpoints. During finetuning, each training sample is paired with a randomly sampled latency budget ranging from 0.5 to 1.0, as by default we only operate on the top half of the layers in LLM. We set the learning rate to 10^{-5} for the original LLaVA model and the scheduler, while keeping other training hyperparameters consistent with the original LLaVA stage-2 finetuning protocol.

Benchmarks and metrics. We conduct a comprehensive evaluation across multiple visual understanding benchmarks, including VQAv2 [14], ScienceQA [39], TextVQA [51], MME [13], and MMBench [38]. We also evaluate on hallucination benchmarks such as POPE [33]. For TextVQA, we specifically focus on the image-based subset, where each question is paired with its corresponding image content. For each benchmark, we report the official metrics on the same dataset splits as in LLaVA-1.5. We report accuracy for VQAv2, ScienceQA, TextVQA and MMBench, perception score for MME, and F1 score for POPE. Additionally, we consider varying latency budgets (from 0.5 to 1.0) when evaluating AdaLLaVA. We report the Prefill FLOPs and time on MME benchmark.

Baselines and model variants. We mainly compare our model with base model LLaVA-1.5 [36]. We evaluate AdaLLaVA with 7B and 13B (see supplement) models, and across two different designs: (a) AdaLLaVA-L for selecting Transformer blocks; and (b) AdaLLaVA-H for selecting attention heads and MLP activations. In our additional analyses, we also use Mipha-3B [70] as the base model.

4.2. Main Results

Comparison to baselines. Our main results across six benchmarks are summarized in Tab. 1. AdaLLaVA demon-

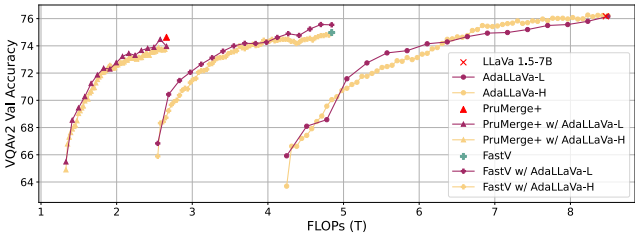

Figure 3. **Accuracy-latency tradoffs of AdaLLaVA** with LLaVA-1.5-7B and additional token selection techniques (PruMerge+ / FastV). Results reported on VQAv2.

strates competitive performance with notable efficiency improvements across all benchmarks.

AdaLLaVA-L, when applied to LLaVA-1.5 7B, maintains comparable performance under full computational budgets. With reduced compute budgets, AdaLLaVA-L shows minimal performance degradation with an average accuracy drop of only 1.5% at 85% budget and 3.4% at 60% budget. Remarkably, at 60% compute budget, AdaLLaVA-L even has slightly better results than the base model on ScienceQA (66.9 vs. 66.8) and POPE (86.1 vs. 85.9).

AdaLLaVA-H shows similar results, with only 1% average performance drop at 85% budget, and 1.9% at 60% budget. The superior performance of AdaLLaVA-H compared to AdaLLaVA-L can be attributed to its head/neuron-level switching mechanism, allowing for more fine-grained control over computational resources than layer-level switches used in AdaLLaVA-L.

Importantly, for all results, AdaLLaVA adheres to the specified latency budgets (see Sec. 4.3). We provide results on additional VQA benchmarks in Supp. B Tab. A.

Integration with token selection. Token selection techniques have demonstrated recent success in improving the efficiency of MLLMs [8, 49]. AdaLLaVA presents an orthogonal direction in adaptive inference. We now demonstrate that AdaLLaVA can be integrated with token selection to further enhance the efficiency. We combine AdaLLaVA with PruMerge+ [49] and FastV [8], two latest token selection methods designed for MLLMs. For FastV, we set filtering layer K=2 and filtering ratio R=50% to ensure consistent comparison. The results are shown in Tab. 1.

With the integration of PruMerge+ or FastV, AdaLLaVA shows significantly improved efficiency across board, when compared to AdaLLaVA with LLaVA-1.5. Again, AdaLLaVA adapts to varying latency budgets and achieves competitive performance relative to the base model (PruMerge+/FastV). For example, with PruMerge+, AdaLLaVA-H shows 2.45% average performance boost at 85% compute budget and only 1.01% performance drop at 60%. A surprising observation is that AdaLLaVA-H achieves strong performance at 85% latency budget, sometimes beating the base model with token pruning. Overall, our results suggest that AdaLLaVA complements to existing token selection approaches. When integrated with Prumerge+ at an 85% latency budget, our approach reduces computational requirements by 70% while maintaining performance with only 1.7% drop in accuracy.

Figure 4. **Visualization of attention between the input latency token and visual tokens** with a 100% latency budget.

4.3. Additional Analyses

Latency adaptivity. We now evaluate the key capability of AdaLLaVA: its adaptivity to input latency budget, *i.e.*, *the ability to complete inference under varying latency requirements using a single model*. We report the accuracy-latency tradeoff of AdaLLaVA variants (*i.e.*, Pareto curves), both with and without token selection, on the VQAv2 benchmark. These results are shown in Fig. 3.

Our results show that AdaLLaVA can empower a base MLLM with static compute footprint (*i.e.*, LLaVA-1.5, PruMerge+, or FastV as individual dots in the Fig. 3) to adapt to varying accuracy-latency tradeoffs (*i.e.*, the corresponding curves in Fig. 3). With varying latency budgets from 50% to 100%, AdaLLaVA effectively trades compute with accuracy. Integrating with token selection methods (PruMerge+ / FastV) further improves the overall efficiency. Thanks to our sampling process in the probabilistic modeling, AdaLLaVA maintains 0% latency violation. We provide additional visualization of execution plans with different latency in Fig. E in Supp. D.

Content adaptivity. It is worth noting that AdaLLaVA is also adaptive to the input content, *i.e.*, *with the same latency budget, its execution plan is dynamically adjusted based on input*. While not our main focus, we present results to illustrate our model's content adaptivity, with the aim of providing insights into its behavior and aiding in its diagnosis.

We visualize attention maps from the latency token to

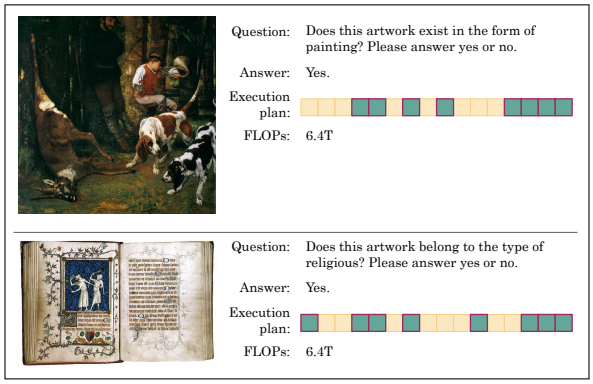

Figure 5. **Visualization of execution plans for different input**. The plan is color-coded with **enable** or **disable** for the 16th to 32th Transformer blocks (left to right). The latency budget is 75%.

all input visual tokens, computed right before the latency token is fed into the scheduler. This is shown in Fig. 4. These attention maps highlight key regions in the input image for answering the target question. For example, in the top, attention concentrates on "Yes Man" for the movie title question but shifts to the actor name for actor identification question. Further, we visualize the execution plans of different input content given by our scheduler in Fig. 5. Under the same latency budget, AdaLLaVA generates distinct execution plans conditioned on the different visual content. These results show AdaLLaVA's ability to dynamically adjust its computational focus based on the input image and text query. See our Supp. D for additional visualizations.

Generalization across MLLMs. We further demonstrate that AdaLLaVA can generalize to other MLLMs beyond LLaVA. We consider Mipha-3B [70], a lightweight MLLM built on Phi-2.7B [23]. Specifically, we apply AdaLLaVA-L on Mipha-3B, following its training strategy [70], and report the results on MME benchmark, shown in Tab. 2. These results have similar trend to those with LLaVA-1.5 in Tab. 1. Complete results are presented in Fig. B in Supp. B.

Ablation: probabilistic vs. deterministic modeling of the scheduler. We present two design choices of the scheduler: deterministic and probabilistic (see Sec. 3.3). For our main results, we adopt the probabilistic version with conditional sampling (detailed in Sec. 3.5). We now compare these two approaches across different latency budgets on the VQAv2 benchmark, using AdaLLaVA-L 7B model. The results are summarized in Tab. 3. Our probabilistic model demonstrates superior adaptability across different latency budgets compared to the deterministic approach. We notice deterministic approach has noticeable performance drop given low latency budget due to under-utilization, and sometimes violates the latency budget. These results confirm our choice of the probabilistic modeling.

Additional ablations. Ablations on (1) the number and granularity of switches; (2) different designs of switches (*i.e.*, AdaLLaVA-H vs. AdaLLaVA-L); and (3) sampling strategies are included in Supp. C due to space limit.

Model	VQA^{v2}	SQA^{I}	VQA^{T}	POPE	MME	MMBench
Mipha-3B	81.3	70.9	56.6	86.7	1488.9	69.7
w/ AdaLLaVA-L-100%	81.1	70.9	55.3	87.7	1450.4	69.2
w/ AdaLLaVA-L-85%	80.4	71.0	53.0	87.8	1429.3	69.0
w/ AdaLLaVA-L-60%	77.2	68.4	44.8	88.0	1397.3	64.6

Table 2. **Generalization** of AdaLLaVA to Mipha-3B.

	AdaLLaVA-L (probabilistic scheduler)			**AdaLLaVA-L** (deterministic scheduler)		
Latency budget	Accuracy	Success (%)	Utilization (%)	Accuracy	Success (%)	Utilization (%)
0.95	75.6	100.0	98.7	75.6	96.1	87.6
0.85	74.9	100.0	99.2	74.6	100.0	80.4
0.75	74.3	100.0	100.0	73.5	100.0	83.2
0.65	72.7	100.0	96.5	72.2	100.0	83.1

Table 3. **Ablation** on deterministic vs. probabilistic modeling for the scheduler. Results reported using 7B model on VQAv2.

5. Conclusion and Discussion

In this paper, we introduced AdaLLaVA, a novel adaptive inference framework designed for MLLMs. AdaLLaVA features a lightweight, learning-based scheduler and a probabilistic modeling technique. It empowers a base MLLM with the ability to adapt to varying latency budgets at inference time. Extensive experiments across benchmarks demonstrated that AdaLLaVA is capable of producing latency- and content-aware execution plans, effectively achieving a range of accuracy-latency tradeoffs.

Adaptive inference of MLLMs. Unlike LLMs, MLLMs include a vision encoder and process a large number of redundant visual tokens. While our paper focuses on the scheduling of the LLM component, this adaptivity can be further extended to token selection and vision encoder. We hope this work will be a step toward making MLLMs more viable for real-world applications where computational resources may be constrained and fluctuate significantly.

Relationship to other efficiency methods. This paper explores adaptive inference in MLLMs, emphasizing adaptability to varying latency budgets within a single model. Our approach is orthogonal to prior methods aimed at improving inference efficiency, such as sparse attention [9] and token selection [49]. Indeed, many of these techniques (*e.g.* token selection as shown in the paper) can be integrated with our framework to further enhance efficiency.

Practical deployment. Our work focuses on algorithm-level innovation, leaving system-level optimization as future work. Conceptually, serving AdaLLaVA is similar to serving MoE-based LLMs [20], which also dynamically routes tokens to different execution paths based on the input. We express compute budgets as percentages of base model FLOPs to abstract hardware/software variations, leaving cross-device portability to future work. We invite joint effort from the vision, learning, and systems communities to further explore these directions.

Acknowledgment. This research was supported in part by the National Science Foundation under Grant Numbers CNS 2333487 / 2333491 (CPS Frontier), CNS 2146449 (CAREER), and IIS 2442739 (CAREER), by the Army Research Lab under contract number W911NF-2020221, and by gift funding from Google and AWS. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] 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

[2] Anthropic. The system card: Claude opus 4 & claude sonnet 4. https://www-cdn.anthropic.com/07b2a3f9902ee19fe39a36ca638e5ae987bc64dd.pdf, 2025. 1

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023. 3

[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] 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. 2

[6] 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. 3

[7] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023. 3

[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, 2024. 1, 3, 6, 7

[9] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 8

[10] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3

[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. 3

[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. 6, 3

[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. 6, 3

[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] Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015. 1

[17] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018. 2

[18] 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

[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] Haiyang Huang, Newsha Ardalani, Anna Sun, Liu Ke, Shruti Bhosale, Hsien-Hsin S. Lee, Carole-Jean Wu, and Benjamin Lee. Toward efficient inference for mixture of experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 8

[21] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019. 2

[22] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 5, 1

[23] 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. 3, 8, 2

[24] 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

[25] 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

[26] 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

[27] 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

[28] Hugo Laurençon. Introducing idefics: An open reproduction of state-of-the-art visual language model. https://huggingface.co/blog/idefics, 2023. 3

[29] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal LLMs with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023. 2

[30] 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, 3

[31] 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

[32] 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, 3

[33] 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. 6, 3

[34] 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, 3

[35] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023. 1, 2

[36] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024. 1, 2, 6, 3

[37] 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

[38] 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. 6, 3

[39] 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. 6, 3

[40] 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

[41] 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

[42] 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

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

[44] 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

[45] 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. 3

[46] 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

[47] 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

[48] 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. 3

[49] 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, 3, 6, 7, 8

[50] 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. 3

[51] 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. 6, 3

[52] 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. 2

[53] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014. 5

[54] 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

[55] 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

[56] 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

[57] A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *NIPS*, 2017. 5

[58] 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. 3

[59] 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, 1

[60] Ran Xu, Jayoung Lee, Pengcheng Wang, Saurabh Bagchi, Yin Li, and Somali Chaterji. Litereconfig: Cost and content aware reconfiguration of video object detection systems for mobile gpus. In *Proceedings*

of the Seventeenth European Conference on Computer Systems, pages 334–351, 2022.

[61] Ran Xu, Fangzhou Mu, Jayoung Lee, Preeti Mukherjee, Somali Chaterji, Saurabh Bagchi, and Yin Li. SmartAdapt: Multi-branch object detection framework for videos on mobiles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2528–2538, 2022. 2

[62] 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

[63] 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

[64] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023. 2

[65] 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. 3

[66] Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*, 2024. 5

[67] Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. AIM: Adaptive inference of multi-modal LLMs via token merging and pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. 2

[68] 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. 3

[69] 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

[70] Minjie Zhu, Yichen Zhu, Xin Liu, Ning Liu, Zhiyuan Xu, Chaomin Shen, Yaxin Peng, Zhicai Ou, Feifei Feng, and Jian Tang. A comprehensive overhaul of multimodal assistant with small language models. *arXiv e-prints*, pages arXiv–2403, 2024. 6, 8, 2

[71] 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. 3