AdaptLink: A Heterogeneity-Aware Adaptive Framework for Distributed MLLM Inference

Xinyi Hu[1], Zihan Chen[2], Kun Guo[3], Meng Zhang[1], Howard H. Yang[1]*

[1]ZJU-UIUC Institute, Zhejiang University
[2]Singapore University of Technology and Design
[3]School of Communications and Electronics Engineering, East China Normal University
xinyih@zju.edu.cn, zihan_chen@mymail.sutd.edu.sg, kguo@cee.ecnu.edu.cn, mengzhang@intl.zju.edu.cn, haoyang@intl.zju.edu.cn

Abstract

Multimodal Large Language Models (MLLMs) have demonstrated exceptional performance in tasks such as commonsense reasoning and visual scene understanding. Despite their success, deploying such models onto resource-constrained edge devices remains challenging due to their cost-intensive properties, while lightweight on-device deployment techniques often compromise performance, which is particularly undesirable for tasks requiring fine-grained generalization. In this paper, we propose *AdaptLink*, a framework that enables a set of edge devices to perform collaborative parallel inference on MLLMs, fully utilizing the available but heterogeneous resources. Unlike prior approaches that uniformly split the model and assign equal computational workload across devices, neglecting the diversity across their computational and hardware conditions, *AdaptLink* dynamically partitions the model into sub-blocks and assigns computing tasks based on device-specific capabilities, accounting for heterogeneous computational power, memory capacity, and inter-device bandwidth. *AdaptLink* achieves desirable inference performance by ensuring parallel execution, minimizing idle time, and optimizing resource utilization. Additionally, the proposed framework incorporates a stability assurance mechanism by pre-loading backup sub-blocks onto inactive devices, aiming to mitigate delay caused by device dropout and redeployments. Extensive experiments on LLaVA-series models demonstrate that compared to baseline methods, *AdaptLink* achieves up to a $1.37\times$ speedup in inference throughput while maintaining model performance. These results underscore the potential of *AdaptLink* as a robust solution for deploying large-scale MLLMs in mobile edge systems, offering efficiency and adaptability in heterogeneous networks.

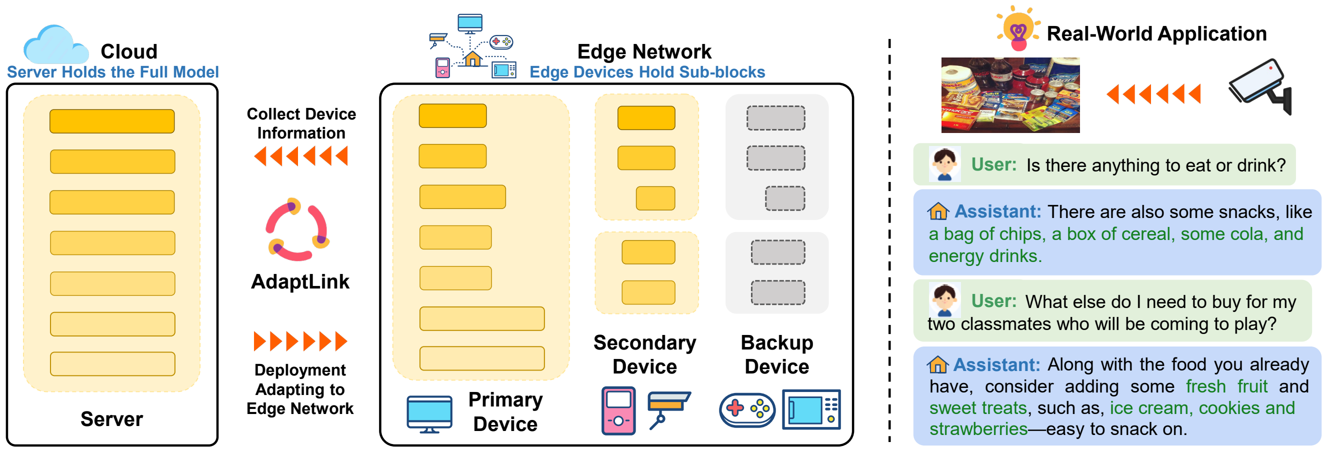
Introduction

The advent of Multimodal Large Language Models (MLLMs) represents a groundbreaking transformation in AI capabilities, augmenting off-the-shelf LLMs (OpenAI 2023; MetaAI 2024) to support seamless integration of vision and language via cost-effective training strategies. The emergence of MLLMs, including LLaVA series (Liu et al. 2023c, 2024, 2023b), BLIP-2 (Li et al. 2023a), and Flamingo (Alayrac et al. 2022), has unlocked new possibilities for multimodal applications. These models can process and analyze complex visual inputs, such as high-resolution images and lengthy videos, and generate coherent and contextually accurate textual outputs. This technological leap has paved the way for applications ranging from autonomous systems (Cui et al. 2024) and smart homes (Li et al. 2024b) to healthcare (Liu et al. 2023a) and entertainment (Ge et al. 2024; Qin et al. 2024). Nonetheless, a critical challenge remains–these advancements also impose significant computational demands (Chen et al. 2024), particularly in tasks involving long video understanding (Ren et al. 2024) or generating detailed and extended textual descriptions from high-resolution visual inputs (Li et al. 2024c). Moreover, deploying MLLMs over edge networks further exacerbates these challenges due to limited computational power, memory, and network bandwidth.

Recent efforts to deploy MLLMs in edge devices could be mainly categorized into two themes. One involves taking existing pre-trained MLLMs and reducing the model size to fit resource-constrained devices through compression techniques such as pruning (Wang et al. 2024b), quantization (Lin et al. 2024; Zhang et al. 2024b), and distillation (Wang et al. 2024a). However, this inevitably incurs performance loss. The second theme involves collaborative edge computing, capitalizing on the substantial resources available on the geo-distributed edge devices. For example, the Cloud-Edge method (Wang et al. 2023; Chen et al. 2023) partitions the model into two parts evenly: one is allocated to the edge device, and another is allocated to the cloud server. While this protects privacy, transferring intermediate activations does require much more than transferring data, introducing more significant latency than deploying the model in the cloud. On the other hand, techniques (Zhang et al. 2024c; Zhao et al. 2024) like EdgeShard underutilized computational resources due to the sequential inference structure. Thus, even when multiple edge devices are available, they cannot be effectively exploited to accelerate inference further. This underscores the need for methods that efficiently utilize multiple edge devices to accelerate inference without compromising performance.

To improve the responsiveness and accuracy of device models within privacy-sensitive environments, we present *AdaptLink*, a framework that enables the distributed deploy-

*Corresponding author


Cloud
Server Holds the Full Model
Server
Collect Device Information
AdaptLink
Deployment Adapting to Edge Network
Edge Network
Edge Devices Hold Sub-blocks
Primary Device
Secondary Device
Backup Device
Real-World Application

User: Is there anything to eat or drink?

Assistant: There are also some snacks, like a bag of chips, a box of cereal, some cola, and energy drinks.

User: What else do I need to buy for my two classmates who will be coming to play?

Assistant: Along with the food you already have, consider adding some fresh fruit and sweet treats, such as, ice cream, cookies and strawberries—easy to snack on.

Figure 1: *AdaptLink* designed for privacy- and latency-sensitive scenarios, embodies a collaborative parallel inference paradigm for heteroogeneous edge networks. As illustrated on the left side of the figure, our approach efficiently utilizes the computational redundancy of local edge devices to enable device-to-device collaborative inference, reducing response time while safeguarding user privacy. Additionally, the inclusion of backup devices enhances the adaptability of MLLMs deployed on edge, allowing them to respond seamlessly to dynamic participation, such as device joining or exiting. On the right is an application example of *AdaptLink*, showcasing its capability to deliver accurate and rapid responses.

ment of MLLMs across heterogeneous edge devices in collaborative manner. As depicted in Figure 1, our approach is built upon tensor parallelism, which divides the transformer block into multiple sub-blocks of different sizes, each assigned to a different device. The core idea is to assign computational tasks according to the device's capability (e.g., computational power, memory capacity, and inter-device bandwidth). For powerful devices (named primary devices), we assign more computationally demanding tasks. On the contrary, weak devices (named secondary devices) afford less computing loads. In this fashion, it ensures all activated devices complete their computations and communication in parallel, minimizing idle time and maximizing resource utilization. Besides, our framework also provides a stability assurance mechanism via preloading backup sub-blocks onto inactive devices (named backup devices) to mitigate the extra delay in the case of device dropout and redeployments.

Note that in our framework, these devices are considered "trusted", allowing less powerful secondary devices to benefit from MLLM services without risking data leakage. For instance, mobile devices shared among various applications (e.g., surveillance systems, smart TVs, and robotic vacuums) or a trusted group of family and friends in a smart home environment can collaboratively process different computational modules of MLLM inference in parallel, thus ensuring both efficiency and privacy security. This collaborative computation also minimizes the need for extensive quantization of model weights while maintaining model performance. To summarize, our paper presents the following contributions:

- We propose a general framework for deploying MLLMs in edge networks, enabling collaborative inference across heterogeneous edge devices.
- Our method classifies available devices into primary and secondary groups based on computational capabilities, memory, and bandwidth. By dynamically partitioning the model, our method achieves balanced workload distribution and maximizes resource utilization.
- We introduce a mechanism to preload backup sub-blocks onto inactive devices, effectively reducing latency caused by device dropouts or redeployments.
- Our method achieves up to $1.37\times$ speedup in inference throughput on LLaVA-series models while preserving model performance.

Related Work

MLLMs

MultiModal pre-training research has witnessed significant advancements in recent years (Zhang et al. 2024a), consistently pushing performance boundaries across a spectrum of downstream tasks (Li et al. 2020; Wang et al. 2022; Zellers et al. 2022; Cai et al. 2024; Ye et al. 2024). The debut of GPT-4 (Vision) (OpenAI 2023) and Gemini (Team et al. 2023) marked a turning point, showcasing impressive multimodal understanding and generation capabilities and sparking a research fervor on MLLMs. Early research primarily focused on multimodal content comprehension and text generation. For instance, BLIP-2 (Li et al. 2023a), LLaVA (Liu et al. 2023c), MiniGPT-4 (Zhu et al. 2023), and OpenFlamingo (Awadalla et al. 2023) advanced image-text understanding, while works like VideoChat (Li et al. 2023b), Video-ChatGPT (Maaz et al. 2023), and LLaMA-VID (Li, Wang, and Jia 2025) extended these capabilities to video-text understanding. Subsequently, the functionality of MLLMs has expanded to support specific modality outputs. For example, GILL (Koh, Fried, and Salakhutdinov 2024), MiniGPT-5 (Zheng, He, and Wang 2023), and

SpeechGPT (Zhang et al. 2023) targeted visual, textual, and speech generation, respectively. Recent efforts have sought to mimic human-like any-to-any modality conversion, as exemplified by Visual ChatGPT (Wu et al. 2023), HuggingGPT (Shen et al. 2024), and AudioGPT (Huang et al. 2024). These advancements shed light on the path to artificial general intelligence. However, as the scale of models and datasets expands, the excessive computational costs of MLLMs impede their deployment, particularly in resource-constrained environments. Existing methods like pruning (Wang et al. 2024b), quantization (Lin et al. 2024), and distillation (Wang et al. 2024a) trade off efficiency for performance, while collaborative edge computing approaches often underutilize available resources (Zhang et al. 2024c; Zhao et al. 2024).

To address these computational challenges, this work introduces a new paradigm for on-device MLLM deployment. By leveraging parallelism across multiple heterogeneous devices, our approach ensures efficient resource utilization while maintaining model performance. This represents a critical step toward enabling MLLM inference in resource-constrained settings.

On-Device Deployment

MLLMs are both computationally intensive and memory-demanding, presenting significant challenges for deployment in resource-constrained environments. To address the *memory wall* problem, quantization has emerged as a widely adopted technique (Xie et al. 2024; Jin et al. 2024). For instance, AWQ (Lin et al. 2024) refines the quantization process by prioritizing the preservation of salient weights, while SmoothQuant (Xiao et al. 2023) mitigates activation outliers through per-channel smoothing factors, facilitating more effective activation quantization. Despite these advances, quantized MLLMs still face two critical limitations: (1) the computational power and memory capacity of a single device remain insufficient for handling large-scale models, and (2) the performance of quantized models often falls short of their full-scale counterparts. To overcome these limitations, other methods have explored distributed computation across multiple devices. PrivateLoRA (Wang et al. 2023) distributes computations between cloud servers and edge devices, while EdgeShard (Zhang et al. 2024c) partitions models layer-by-layer, assigning them across edge devices. Similarly, LinguaLinked (Zhao et al. 2024) enhances system stability by repeatedly deploying model layers on devices. However, these approaches are often constrained by sequential inference or fine-tuning, leading to underutilization of device capabilities and increased communication latency.

In contrast, our proposed inference framework introduces a novel approach by leveraging adaptive parallel computation to fully exploit the arithmetic capabilities of heterogeneous edge devices. By dynamically adjusting the computational workload assigned to each device, our method minimizes the additional latency incurred by inter-device communication, offering an efficient and scalable solution for deploying MLLMs in edge environments.

System Model

In this section, we detail our system model for deploying MLLMs in distributed edge networks with powerful device, two secondary and backup devices as example, clarifying key objective that guides the optimization direction. We identify existing challenges and potential improvements within these systems and establish the performance metrics used throughout this paper. Notably, this system model is versatile, supporting a range of complex reasoning tasks based on the MLLM backbone, including those involving image, video, and multimodal representations.

Consider a distributed edge network comprising a global server and D edge devices. In this setup, the global server partitions the MLLM into K sub-blocks—typically by layer (Zhang et al. 2024c; Zhao et al. 2024)—and allocates them to selected devices for collaborative inference, ensuring user privacy by not sharing any data with the server. The devices exhibit heterogeneous computational and memory capabilities, with the global server possessing significantly greater computational resources than the edge devices. The memory wall is denoted by $\mathcal{M}_{\text{ava}}$, and the required memory of the sub-block is denoted by $\mathcal{M}_{\text{req}}$. Besides, we use a binary variable $H_{k,d}$ to indicate whether sub-block k is allocated to device d. The goal of all the entities is to minimize the overall inference latency, formally expressed as:

$$\min \left(\sum_{k=1}^{K} \sum_{d=1}^{D} T_{\text{comp}}^{k,d} H_{k,d} + \sum_{k=1}^{K} \sum_{d=1}^{D} \sum_{d'=1}^{D} T_{\text{comm}}^{k,d,d'} H_{k,d} H_{k,d'} \right)$$
$$\text{s.t.} \quad \sum_{k=1}^{K} \mathcal{M}_{\text{req}}^{k} H_{k,d} \leq \beta \mathcal{M}_{\text{ava}}^{d}, \quad \forall d \in \{1, \ldots, D\}, \tag{1}$$

where the component $T_{\text{comp}}^{k,d}$ is defined as follows:

$$T_{\text{comp}}^{k,d} = \frac{\mathcal{F}_k}{\mathcal{P}_d}, \tag{2}$$

where $\mathcal{F}_k$ denotes the number of floating-point operations (FLOPs) for sub-block k, and $\mathcal{P}_d$ represents the FLOPs per second (FLOP/s) of device d. The other component $T_{\text{comm}}^{k,d,d'}$ is defined as:

$$T_{\text{comm}}^{k,d,d'} = \begin{cases} \frac{\mathcal{O}_k}{\mathcal{B}_{d,d'}}, & \text{if } d \neq d' \\ 0, & \text{if } d = d', \end{cases} \tag{3}$$

where $\mathcal{B}$ represents the bandwidth between two devices, and $\mathcal{O}$ denotes the size of the sub-blocks' output.

Notably, most existing research partition models by layers to perform sequential inference (Zhang et al. 2024c; Zhao et al. 2024) or partition and assign models by tensors uniformly (Xu and You 2023; Li et al. 2023c), enabling large model functions to operate locally on resource-constrained devices. However, from a system perspective, sequential reasoning is inefficient because it underutilizes device resources by keeping other devices idle during individual computations. As a result, this approach falls short in optimizing response time, resource utilization, and adaptability, particularly in scenarios with significant differences in the computational power of edge devices.

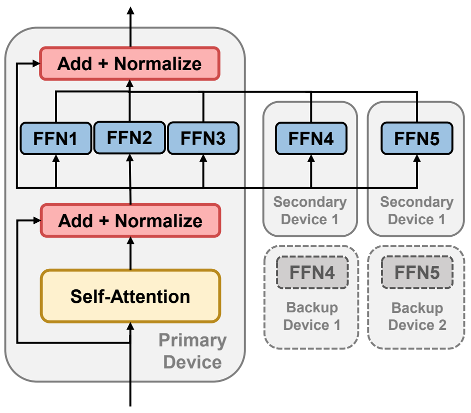
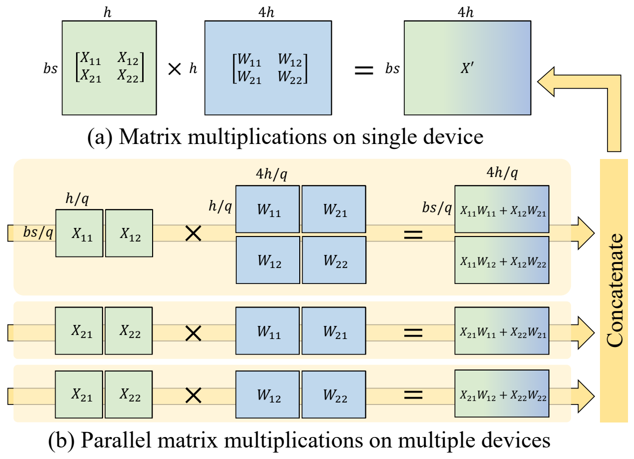


Figure 2: Illustration of transformer block partitioning, using the FFN layer as an example.



h
4h
4h
bs
× h
= bs
X'
(a) Matrix multiplications on single device
h/q
4h/q
4h/q
bs/q
h/q
bs/q
Concatenate
(b) Parallel matrix multiplications on multiple devices

Figure 3: Illustration of the 2D non-uniform partition.

To reduce model response time, a natural approach is to minimize inter-device communication time and enable parallel computation across multiple devices. Therefore, in this paper, we propose the *AdaptLink* framework, which maximizes computation on capable devices (i.e., primary devices) and delegates less complex computations to secondary devices to enable parallel processing. This approach requires transmitting only a few parameters, enhancing collaborative inference efficiency. Additionally, *AdaptLink* deploys redundant blocks on inactive devices to ensure fault tolerance against secondary device dropout, thus enhancing the overall resilience of the framework.

AdaptLink Framework

In contexts such as smart cities, autonomous vehicles, and industrial IoT, edge networks demand both high responsiveness and adaptability, particularly given the dynamic nature of edge devices, which can unpredictably disconnect from or rejoin the network. These dynamic conditions necessitate a robust framework that can seamlessly manage and distribute computational workloads across available devices while maintaining low-latency responses. *AdaptLink* introduces a novel approach that harnesses the collective computational resources of edge devices, effectively balancing communication overhead with processing power to optimize efficiency and responsiveness. Its key components include:

- Edge Network Profiling: The cloud server gathers critical metrics from the edge network, such as device memory constraints, computational capacities, and inter-device bandwidths.
- Model Partitioning: The model is divided into sub-blocks based on a predefined strategy for parallel computation.
- Dynamic Allocation: *AdaptLink* allocates sub-blocks to edge devices tailored to each device's capabilities and the network's overall resource distribution.
- Stability Assurance: A stability mechanism dynamically adapts to changes in the edge network, ensuring consistent performance even under fluctuating device availability or capacity.

Model Partition for Parallel Inference

The backbone of MLLMs typically comprises multiple transformer layers stacked sequentially. Each transformer layer contains a self-attention layer and a feed-forward network (FFN) layer. These two components have comparable parameter sizes and share identical input-output activation shapes, represented as $[b, s, h]$, where h denotes the hidden state size, b the batch size, and s the sequence length. In this section, we introduce a generalized 2D partitioning strategy (Xu and You 2023) to optimize layer partitioning for parallel inference.

Notably, approximately 90% of the computation within a transformer block is concentrated in the FFN layer, making it a particularly compelling target for parallel execution. While this section focuses on the FFN layer illustrated in Figure 2 shown. The same partitioning methodology can be applied to the self-attention layer.

We employ a 2D partitioning strategy (Xu and You 2023) to accelerate the inference of extremely large-scale MLLMs. The FFN layer consists of two matrix multiplications separated by a non-linear activation function. In the first matrix multiplication, we compute the product $X' = W \times X$, as Figure 3 (a) shown, where $X \in \mathbb{R}^{bs \times h}$ represents the input and $W \in \mathbb{R}^{h \times 4h}$ denotes the weight matrix.

To enable parallel processing, the weight and input matrices are uniformly divided into $q \times q$ sub-blocks, resulting in submatrices W_{ij}, X_{ij}, and X'_{ij}, where $i, j \in \{1, \dots, q\}$. The computational work of these submatrices will be subsequently distributed according to the capacity of the devices. The matrix multiplications are then executed in parallel across these sub-blocks, as Figure 3 (b) shown. For each sub-block k, the k-th column of W and the k-th row of X are broadcast across their respective rows and columns, allowing distributed processors to compute partial products independently. The partial product for each submatrix is calculated as:

$$X'_{ij} = \sum_{k=1}^{q} W_{ik} \times X_{kj}. \tag{4}$$

This block-wise computation, combined with strategic broadcasting, minimizes inter-processor communication by ensuring that necessary data is localized within each processor after the broadcast. Once all partial products are computed, the final matrix X' is reconstructed by aggregating the results from all processors. The second matrix multiplication follows the same partitioning and computation methodology, ensuring consistent efficiency throughout the FFN layer. Overall, the 2D non-uniform partition strategy introduces some communication overhead but significantly reduces the memory and computational demands on individual devices, thereby lowering the barriers to on-device model deployment.

Dynamic Allocation Strategy

Previous parallel approaches (Li et al. 2024a; Shoeybi et al. 2019; Xu and You 2023) generally assign tensors equally across devices, as these approaches are primarily designed for high-performance cluster environments. However, such strategies are unsuitable for heterogeneous edge networks, where device capabilities, such as computation speed and memory, vary significantly and require tailored allocations to maximize efficiency. To address this issue, *AdaptLink* deploys as many sub-blocks as possible on high-capability devices (primary devices), while less powerful devices (secondary devices) serve in an auxiliary computation role. *AdaptLink* implements a layer-by-layer allocation strategy, beginning with the last transformer layer and iterating through each layer in sequence. For each layer, *AdaptLink* assesses whether the primary device has sufficient memory to perform the inference. If memory is insufficient, a portion of the FFN's computation is offloaded to the secondary device. Once a layer is allocated entirely to the primary device, no further divisions are made for preceding layers.

In a transformer block, the computationally intensive FFN layer is offloaded to a secondary device, while the remaining computations are retained on the primary device. The proportion of FFN computations handled by the primary device is denoted as p, with $1-p$ representing the portion executed by the secondary device. The primary device's computation time for the FFN layer is thus $p \cdot \frac{\mathcal{F}}{\mathcal{P}_{\text{pri}}}$. To complete the sub-blocks' computation, the primary device must wait for the secondary device to finish its auxiliary calculations, resulting in a waiting time of $(1-p)\left(\frac{\mathcal{F}}{\mathcal{P}_{\text{sec}}}+\frac{\mathcal{O}}{\mathcal{B}}\right)$. Since both devices need to complete their respective tasks before progressing to the next computation stage, p is adjusted to balance the load, aiming for both devices to finish their tasks in approximately the same time to minimize idle periods. The optimal value of p could be obtained by solving the following optimization problem:

$$\arg\min_{p}\left|p\frac{\mathcal{F}}{\mathcal{P}_{\text{pri}}}-(1-p)\left(\frac{\mathcal{F}}{\mathcal{P}_{\text{sec}}}+\frac{\mathcal{O}}{\mathcal{B}}\right)\right| \tag{5}$$

This approach ensures optimal load balancing between primary and secondary devices, enhancing overall efficiency. This allocation policy is determined in the cloud, and once sub-block parameters are deployed to edge devices, they remain unchanged.

Stability Assurance Mechanism

To improve the stability and robustness of our proposed framework in the context of potential device dropout, we adopt a stability assurance mechanism. In this mechanism, sub-blocks on secondary devices are duplicated onto the inactive devices to enhance system resilience. If a secondary device fails, an inactive device with the same sub-block can be activated to perform identical computation tasks. When a new device joins the network, it is allocated unassigned sub-blocks and evaluated against the performance of the original device. If the new device demonstrates superior performance, it is promoted to secondary status, while the original device is relegated to a backup role. This mechanism allows computationally stronger devices to replace weaker standby devices, thereby maximizing resource utilization across the edge network and ensuring system stability in response to dynamic device changes.

Experiments

In this section, we first detail our experimental setup, including the baseline models and evaluation protocol. Next, we present the primary results, focusing on both quality and efficiency. We further discuss related studies to provide context and deeper insights into our work. Finally, we validate the rationale behind each design choice.

Experimental Setups

Model. Our method relies on off-the-shelf pretrained models, wherein we primarily experiment with the state-of-the-art open-source image-to-text model, LLaVA (Liu et al. 2024). LLaVA aligns a visual encoder with a large language model via a lightweight projection layer, enabling seamless image-text interaction. It uses a pretrained language model as its backbone to generate coherent, context-aware outputs. Compared to traditional vision-language models, LLaVA integrates richer cross-modal attention and fine-tunes its alignment with carefully curated multimodal datasets, resulting in a more robust and versatile model. However, these improvements increase computational complexity and resource demands, limiting its suitability for resource-constrained or latency-sensitive scenarios.

Baselines. We compare our framework against the following baselines in terms of both quality and efficiency:

- Edge-Only: This baseline deploys the entire model on a single edge device without partitioning, relying solely on the computational resources of the local device. If the model's memory requirements exceed the device's memory, quantization (Xiao et al. 2023) is applied to compress the model.
- Cloud-Only (Liu et al. 2023c): In this configuration, the model resides entirely on a cloud server, with edge devices transmitting data to the cloud for inference and receiving the results.
- Layer-Partitioned (Zhang et al. 2024c; Gugger et al. 2022): This approach partitions the model layer-by-layer across multiple devices, executing inference sequentially through the designated layers on each device.

Table 1: Throughput (tokens/s) comparison across different model scales.

Method	LLaVA-7B		LLaVA-13B		LLaVA-34B	
	Value	Speedup	Value	Speedup	Value	Speedup
Cloud-Only	10.95	-	8.5	-	4.9	-
Edge-Only	**14.95**	**1.37**×	0.82 (8-bit)	0.096×	**7.73** (4-bit)	**1.57**×
Layer-Partitioned	10.65	0.97×	7.74	0.9×	4.7	0.95×
Ours	**14.98**	**1.37**×	**9.29**	**1.09**×	5.88	1.2×

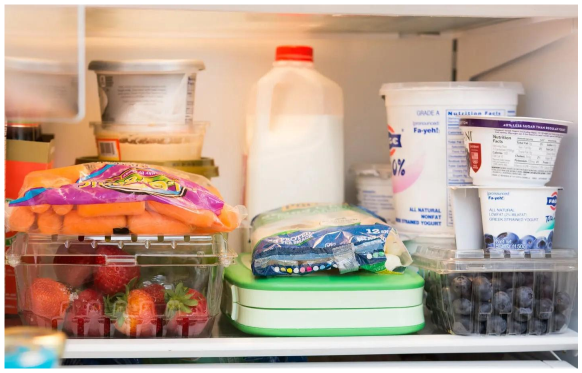
Metrics. This study focuses on image-to-text tasks where the number of tokens generated in each response varies, latency is not a representative metric. Instead, we evaluate it in terms of text generation quality and throughput, measured as the number of tokens generated per second (tokens/s). This provides a more reliable assessment of the system's efficiency in handling variable-length outputs.
Implementation details. We simulate the collaborative edge network using five 4090 GPUs, each with different memory constraints. One GPU, designated as the primary device, is allocated the full 24GB of GPU memory, while the remaining four GPUs, serving as secondary or badckup devices, are evenly split, with two allocated 12GB and the other two allocated 6GB of GPU memory, respectively.

Main Results

Compared to the theoretical computation reduction, on-hardware acceleration is more critical for real-world applications. To demonstrate the effectiveness of our method, we report the end-to-end throughput and speedups in Table 1. When deploying the smaller-scale LLaVA-7B model, our approach demonstrates a significant acceleration compared to the Cloud-Only and Layer-Partition approaches. This is primarily due to the limited bandwidth (typically 1Mbps) for transferring images, which range in size from a few hundred kilobytes to several megabytes, from the local device to the cloud. In the Cloud-Only approach, the majority of the latency is caused by the communication with the cloud server. In contrast, the delay in the Layer-Partitioned method arises from the intermediate activation communication required during its sequential computation. Because the Layer-Partitioned approach does not dynamically adjust the model partitioning process, it unnecessarily offloads the model to secondary devices even when the primary device is capable of handling the full model, thus increasing the communication time. The Edge-Only, like our approach in the current setup, involves deploying the entire model on the primary device, avoiding any additional communication overhead, and achieving a speedup of 1.37×.

As the model size increases, the advantages of our parallelized computation strategy over alternative methods become more evident. Specifically, for LLaVA-13B, the Edge-Only method cannot meet the model's larger memory requirements, necessitating quantization, which introduces

Challenging example from LLaVA-Bench (In-the-Wild)

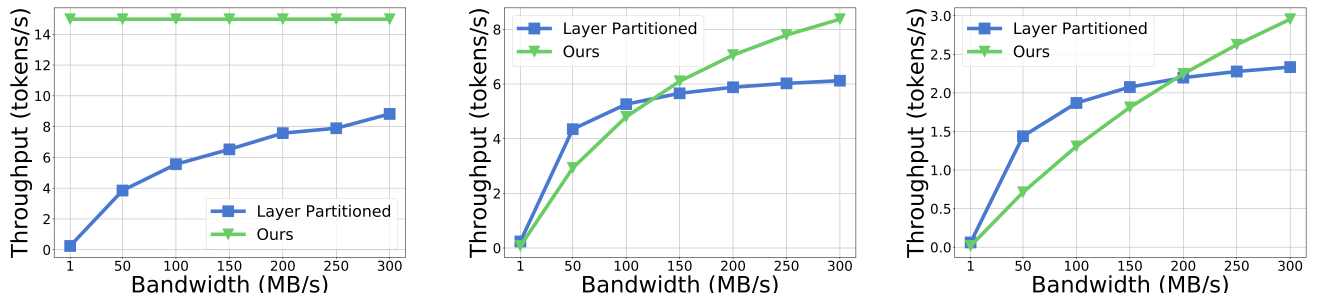


Question 1	Is there strawberry-flavored yogurt in the fridge?
Ours	Yes, there is a container of strawberry-flavored yogurt in the fridge. It's located on the top shelf, to the right of the milk.
Edge-Only (4-bit)	Yes, there is a container of strawberry-flavored yogurt in the fridge. It's on the top shelf, to the right of the milk.
Question 2	What is the brand of the yogurt?
Ours	The brand of the blueberry-flavored yogurt is Yoplait.
Edge-Only (4-bit)	The brand of the blueberry-flavored yogurt is "Fage".

Table 2: Example prompt from the LLaVA (Liu et al. 2024) paper for comparing the capabilities of visual reasoning and capabilities. Compared to the 4-bit quantized LLaVA-34B model deployed with the Edge-Only method, the LLaVA-34B model deployed using our method demonstrates a more accurate understanding of the image and performs better on challenging samples..

performance bottlenecks. This is particularly true for devices like the NVIDIA 4090 GPU used in our experiments, where the lack of hardware support for quantization acceleration leads to slower inference despite reduced memory usage. In contrast, the Layer-Partitioned suffers from the inefficiencies of sequence inference, resulting in slower performance compared to our parallelized computation strategy.

When deploying the larger LLaVA-34B model, we had to use 4-bit quantization, as 8-bit quantization failed to meet the memory requirements for the Edge-Only deployment. Notably, the NVIDIA 4090 GPUs demonstrated better support for 4-bit quantization, resulting in faster inference speeds that even exceeded those of our approach. This highlights that the deployment of models on edge devices depends on their support for specific algorithms, and even widely-used quantization techniques are not supported on all devices. It is also well-established that higher levels of quantization lead to greater performance degradation. The



(a) LLaVA-7B (b) LLaVA-13B (c) LLaVA-34B

Figure 4: Impact of Bandwidth to Throughput.

4-bit quantized model, while faster in inference, suffers from reduced performance in graph image comprehension, as demonstrated in Question 2 of Table 2.

The results of the above experiments demonstrate that our method achieves superior throughput for larger models by efficiently distributing computation across multiple devices. This approach mitigates the drawbacks associated with quantization and sequence inference, showcasing the effectiveness of our method in handling large-scale LLaVA models while optimizing both throughput and computational efficiency

Impact of Bandwidth

This section investigates the impact of network bandwidth on performance. Since the performance of Cloud-Only and Edge-Only methods are unaffected by inter-device communication bandwidth, we only compare our method with the distributed inference approach, i.e., Layer-Partitioned.

As shown in Figure 4 (a), for LLaVA-7B, our method can deploy the entire model on the primary device since the available memory is sufficient, thus eliminating the need for inter-device communication. This results in significantly higher throughput compared to the Layer-Partitioned approach, and the performance is independent of local bandwidth. For LLaVA-13B and LLaVA-34B, however, due to memory constraints, we must employ a distributed deployment. As illustrated in Figures 4 (b) and (c), our method begins to demonstrate its advantages when the bandwidth exceeds 150MB, as the additional communication overhead is offset by improved computational efficiency.

Stability Evaluation

To assess the impact of unexpected device dropout, we simulate the random disconnection of a device within the edge network and perform a single inference. The results reveal distinct behaviors based on the role of the disconnected device. When the primary device is dropped, the throughput of our method decreases to 3.02 tokens/s, compared to 2.9 tokens/s for the Layer-Partition approach, reflecting an approximate 50% reduction. This performance decline occurs because the primary device lacks backup and requires reloading the model, with the model loading time being comparable to the inference time, both approximately 20 seconds.

In contrast, when a secondary/backup device is disconnected, our method maintains a throughput of 5.71 tokens/s with negligible degradation, while the Layer-Partition approach experiences a 50% reduction, dropping to 2.86 tokens/s. This stability in our method is attributed to the deployment of standby modules capable of seamlessly replacing dropped devices. These findings demonstrate that our approach effectively adapts to dynamic changes within the edge network, ensuring robust and stable performance.

Limitation and Discussion

The results indicate that our method provides a significant advantage in high-bandwidth scenarios, such as LANs of various institutions, and similar environments. However, in low-bandwidth scenarios, our approach performs less favorably compared to the Layer-Partitioned method, primarily due to frequent communication overheads. To address this, we plan to explore the integration of an asynchronous communication strategy (Li et al. 2024a), which aims to mitigate the impact of communication latency by overlapping communication with computation, enabling better adaptation to various bandwidth environments

Conclusion

In this paper, we proposed *AdaptLink*, a method to deploy MLLMs on heterogeneous edge networks through parallelism. Our approach partitions the model into multiple subblocks and assigns them to edge devices based on their available memory, computational capability, and network bandwidth. Our method achieves the lossless deployment of the large-scale LLaVA series model at the edge while improving throughput, especially in high bandwidth scenarios. This advancement offers a novel perspective for the future deployment of generative AI applications in real-world scenarios.

References

Alayrac, J.-B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35: 23716–23736.

Awadalla, A.; Gao, I.; Gardner, J.; Hessel, J.; Hanafy, Y.; Zhu, W.; Marathe, K.; Bitton, Y.; Gadre, S.; Sagawa, S.; et al. 2023. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*.

Cai, M.; Liu, H.; Mustikovela, S. K.; Meyer, G. P.; Chai, Y.; Park, D.; and Lee, Y. J. 2024. Making Large Multimodal Models Understand Arbitrary Visual Prompts. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, Y.; Li, R.; Zhao, Z.; Peng, C.; Wu, J.; Hossain, E.; and Zhang, H. 2023. Netgpt: A native-ai network architecture beyond provisioning personalized generative services. *arXiv preprint arXiv:2307.06148*.

Chen, Z.; Yang, H. H.; Tay, Y. C.; Chong, K. F. E.; and Quek, T. Q. S. 2024. The Role of Federated Learning in a Wireless World with Foundation Models. *IEEE Wireless Communications*, 31(3): 42–49.

Cui, C.; Ma, Y.; Cao, X.; Ye, W.; Zhou, Y.; Liang, K.; Chen, J.; Lu, J.; Yang, Z.; Liao, K.-D.; et al. 2024. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 958–979.

Ge, Z.; Huang, H.; Zhou, M.; Li, J.; Wang, G.; Tang, S.; and Zhuang, Y. 2024. Worldgpt: Empowering llm as multimodal world model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 7346–7355.

Gugger, S.; Debut, L.; Wolf, T.; Schmid, P.; Mueller, Z.; Mangrulkar, S.; Sun, M.; and Bossan, B. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Huang, R.; Li, M.; Yang, D.; Shi, J.; Chang, X.; Ye, Z.; Wu, Y.; Hong, Z.; Huang, J.; Liu, J.; et al. 2024. Audiogpt: Understanding and generating speech, music, sound, and talking head. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 23802–23804.

Jin, Y.; Li, J.; Liu, Y.; Gu, T.; Wu, K.; Jiang, Z.; He, M.; Zhao, B.; Tan, X.; Gan, Z.; et al. 2024. Efficient multimodal large language models: A survey. *arXiv preprint arXiv:2405.10739*.

Koh, J. Y.; Fried, D.; and Salakhutdinov, R. R. 2024. Generating images with multimodal language models. *Advances in Neural Information Processing Systems*, 36.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, 19730–19742. PMLR.

Li, K.; He, Y.; Wang, Y.; Li, Y.; Wang, W.; Luo, P.; Wang, Y.; Wang, L.; and Qiao, Y. 2023b. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.

Li, M.; Cai, T.; Cao, J.; Zhang, Q.; Cai, H.; Bai, J.; Jia, Y.; Li, K.; and Han, S. 2024a. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7183–7193.

Li, S.; Liu, H.; Bian, Z.; Fang, J.; Huang, H.; Liu, Y.; Wang, B.; and You, Y. 2023c. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, 766–775.

Li, X.; Yin, X.; Li, C.; Zhang, P.; Hu, X.; Zhang, L.; Wang, L.; Hu, H.; Dong, L.; Wei, F.; et al. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, 121–137. Springer.

Li, X.; Zhang, M.; Geng, Y.; Geng, H.; Long, Y.; Shen, Y.; Zhang, R.; Liu, J.; and Dong, H. 2024b. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18061–18070.

Li, Y.; Wang, C.; and Jia, J. 2025. Llama-vid: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, 323–340. Springer.

Li, Z.; Yang, B.; Liu, Q.; Ma, Z.; Zhang, S.; Yang, J.; Sun, Y.; Liu, Y.; and Bai, X. 2024c. Monkey: Image resolution and text label are important things for large multi-modal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 26763–26773.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100.

Liu, F.; Zhu, T.; Wu, X.; Yang, B.; You, C.; Wang, C.; Lu, L.; Liu, Z.; Zheng, Y.; Sun, X.; et al. 2023a. A medical multimodal large language model for future pandemics. *NPJ Digital Medicine*, 6(1): 226.

Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2023b. Improved Baselines with Visual Instruction Tuning.

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

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023c. Visual Instruction Tuning.

Maaz, M.; Rasheed, H.; Khan, S.; and Khan, F. S. 2023. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*.

MetaAI. 2024. Introducing Meta Llama 3: The most capable openly available LLM. *Meta AI Blog*.

OpenAI. 2023. GPT-4 Technical Report.

Qin, Y.; Zhou, E.; Liu, Q.; Yin, Z.; Sheng, L.; Zhang, R.; Qiao, Y.; and Shao, J. 2024. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16307–16316. IEEE.

Ren, S.; Yao, L.; Li, S.; Sun, X.; and Hou, L. 2024. Timechat: A time-sensitive multimodal large language model for long video understanding. In *Proceedings of* *the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14313–14323.

Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; and Zhuang, Y. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; Millican, K.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Wang, G.; Liu, J.; Li, C.; Zhang, Y.; Ma, J.; Wei, X.; Zhang, K.; Chong, M.; Zhang, R.; Liu, Y.; et al. 2024a. Cloud-Device Collaborative Learning for Multimodal Large Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12646–12655.

Wang, M.; Zhao, Y.; Liu, J.; Chen, J.; Zhuang, C.; Gu, J.; Guo, R.; and Zhao, X. 2024b. Large Multimodal Model Compression via Iterative Efficient Pruning and Distillation. In *Companion Proceedings of the ACM on Web Conference 2024*, 235–244.

Wang, P.; Yang, A.; Men, R.; Lin, J.; Bai, S.; Li, Z.; Ma, J.; Zhou, C.; Zhou, J.; and Yang, H. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International conference on machine learning*, 23318–23340. PMLR.

Wang, Y.; Lin, Y.; Zeng, X.; and Zhang, G. 2023. Privatelora for efficient privacy preserving llm. *arXiv preprint arXiv:2311.14030*.

Wu, C.; Yin, S.; Qi, W.; Wang, X.; Tang, Z.; and Duan, N. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Xiao, G.; Lin, J.; Seznec, M.; Wu, H.; Demouth, J.; and Han, S. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 38087–38099. PMLR.

Xie, J.; Zhang, Y.; Lin, M.; Cao, L.; and Ji, R. 2024. Advancing Multimodal Large Language Models with Quantization-Aware Scale Learning for Efficient Adaptation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 10582–10591.

Xu, Q.; and You, Y. 2023. An efficient 2d method for training super-large deep learning models. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 222–232. IEEE.

Ye, Q.; Xu, H.; Ye, J.; Yan, M.; Hu, A.; Liu, H.; Qian, Q.; Zhang, J.; and Huang, F. 2024. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13040–13051.

Zellers, R.; Lu, J.; Lu, X.; Yu, Y.; Zhao, Y.; Salehi, M.; Kusupati, A.; Hessel, J.; Farhadi, A.; and Choi, Y. 2022. Merlot reserve: Neural script knowledge through vision and language and sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16375–16387.

Zhang, D.; Li, S.; Zhang, X.; Zhan, J.; Wang, P.; Zhou, Y.; and Qiu, X. 2023. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. *arXiv preprint arXiv:2305.11000*.

Zhang, D.; Yu, Y.; Li, C.; Dong, J.; Su, D.; Chu, C.; and Yu, D. 2024a. Mm-llms: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*.

Zhang, H.; Ji, X.; Chen, Y.; Fu, F.; Miao, X.; Nie, X.; Chen, W.; and Cui, B. 2024b. Pqcache: Product quantization-based kvcache for long context llm inference. *arXiv preprint arXiv:2407.12820*.

Zhang, M.; Cao, J.; Shen, X.; and Cui, Z. 2024c. EdgeShard: Efficient LLM Inference via Collaborative Edge Computing. *arXiv preprint arXiv:2405.14371*.

Zhao, J.; Song, Y.; Harris, I.; Jyothi, S. A.; et al. 2024. LinguaLinked: Distributed Large Language Model Inference on Mobile Devices. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 160–171.

Zheng, K.; He, X.; and Wang, X. E. 2023. Minigpt-5: Interleaved vision-and-language generation via generative vokens. *arXiv preprint arXiv:2310.02239*.

Zhu, D.; Chen, J.; Shen, X.; Li, X.; and Elhoseiny, M. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.