ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάλυση Απόδοσης και Βελτιστοποίηση της Εξυπηρέτησης Αιτημάτων σε Πολυτροπικά Μεγάλα Γλωσσικά Μοντέλα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών «Επιστήμη Δεδομένων και Μηχανική Μάθηση»

Ιωάννης Π. Δαλιάνης

Αθήνα, Νοέμβριος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάλυση Απόδοσης και Βελτιστοποίηση της Εξυπηρέτησης Αιτημάτων σε Πολυτροπικά Μεγάλα Γλωσσικά Μοντέλα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών «Επιστήμη Δεδομένων και Μηχανική Μάθηση»

Ιωάννης Π. Δαλιάνης

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Νοεμβρίου 2025

. . . . .	. . . . .	. . . . .
Θάλεια Δήμητρα Δούδαλη Επίκουρη Καθηγήτρια στο IMDEA Software Institute	Νεκτάριος Κοζύρης Καθηγητής ΕΜΠ	Γεώργιος Γκούμας Καθηγητής ΕΜΠ

Αθήνα, Νοέμβριος 2025

NATIONAL TECHINCAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF COMPUTER SCIENCE

Performance Analysis and Optimization of Inference Serving for Multimodal Large Language Models

DIPLOMA THESIS
Interdepartmental Program of Postgraduate Studies (Master's Degree Program) "Data Science and Machine Learning"

Ioannis P. Dalianis

Athens, November 2025

NATIONAL TECHINCAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF COMPUTER SCIENCE

Performance Analysis and Optimization of Inference Serving for Multimodal Large Language Models

DIPLOMA THESIS
Interdepartmental Program of Postgraduate Studies (Master's Degree Program) "Data Science and Machine Learning"

Ioannis P. Dalianis

Supervisor: Nectarios Koziris
Professor NTUA

Approved by the three-member examination committee on the 7th of November 2025

. . . . .	. . . . .	. . . . .
Thaleia Dimitra Doudali	Nectarios Koziris	Georgios Goumas
Assistant Professor at IMDEA Software Institute	Professor NTUA	Professor NTUA

Athens, November 2025

. . . . . .

Ιωάννης Π. Δαλιάνης

to my family

Η στοιχειοθεσία του κειμένου έγινε με το XɜTEX 0.999997.

Χρησιμοποιήθηκαν οι γραμματοσειρές Minion Pro, Myriad Pro και `Consolas`.

Σημείωση: Για τη συγγραφή, τη δομή και τη βελτιστοποίηση μέρους του περιεχομένου της παρούσας διπλωματικής χρησιμοποιήθηκαν εργαλεία Τεχνητής Νοημοσύνης (AI) / Μεγάλα Γλωσσικά Μοντέλα (LLMs).

Περίληψη

Η ανάπτυξη και χρήση μεγάλων Πολυτροπικών Γλωσσικών Μοντέλων (MLLMs) συναντά σημαντικούς περιορισμούς λόγω των υψηλών υπολογιστικών και μνημονικών απαιτήσεών τους, οι οποίες εντείνονται από τη διαρκή αύξηση της προσωρινής μνήμης που καταλαμβάνουν οι υπολογισμοί. Η παρούσα διπλωματική εργασία εξετάζει αυτές τις προκλήσεις μέσα από μια συστηματική μελέτη της μείωσης των μεγεθών των μη-κειμενικών δεδομένων, μιας τεχνικής η οποία μειώνει το κόστος επεξεργασίας διατηρώντας παράλληλα την ποιότητα των αποτελεσμάτων.

Πραγματοποιούμε ανάλυση απόδοσης σε ένα ευρύ σύνολο MLLMs χρησιμοποιώντας το σύστημα Εξυπηρέτησης vLLM. Αξιολογούμε την επίδραση της μείωσης των μεγεθών των μη-κειμενικών δεδομένων στην ταχύτητα, στην μνήμη και στην ποιότητα, σε εργασίες εικόνας και βίντεο.

Για τους φόρτους εργασίας εικόνας, η μείωση του μεγέθους της εικόνας επιτυγχάνει έως και 51% μείωση στο χρόνο απόκρισης και 53% μείωση στην επιβάρυνση μνήμης με ελάχιστη υποβάθμιση της ποιότητας ($\approx -4\%$ απώλεια Ακρίβειας). Για τους φόρτους εργασίας βίντεο, η ανίχνευση αλλαγής σκηνής για έξυπνη δειγματοληψία καρέ προσφέρει ακόμη πιο σημαντικά κέρδη, επιτυγχάνοντας έως και 79% μείωση στον χρόνο και 76% μείωση στην επιβάρυνση μνήμης, διατηρώντας παράλληλα ανταγωνιστική ποιότητα αποτελεσμάτων.

Τα αποτελέσματα αυτά δείχνουν ότι η μείωση των μεγεθών των μη-κειμενικών δεομένων αποτελεί ουσιαστική τεχνική συστημικής βελτιστοποίησης, συμβάλλοντας στη μείωση καθυστέρησης και στην αύξηση της αποδοτικής εξυπηρέτησης MLLMs.

Λέξεις-Κλειδιά

Πολυτροπικά Μεγάλα Γλωσσικά Μοντέλα, Βελτιστοποίηση Εξυπηρέτησης Αιτημάτων, Μείωση Αποτυπώματος KV Cache, Απόδοση Συστημάτων Τεχνητής Νοημοσύνης, Μείωση Μεγεθών μη-Κειμενικών Δεδομένων (Modality Reduction)

Abstract

Deploying large Multimodal Large Language Models (MLLMs) in production is critically bottlenecked by their immense computational and memory demands, stemming from the Transformer's quadratic complexity in the prefill phase and the resulting Key-Value (KV) cache memory exhaustion. This thesis addresses these system challenges by validating modality reduction as a powerful, system-level optimization technique. We perform a rigorous performance analysis using a diverse MLLM set served by the high-throughput engine vLLM on NVIDIA A100 hardware, evaluating the trade-off between speed and task quality across image and video workloads. Our findings quantify the effectiveness of two primary reduction strategies. For image workloads, proportional image downsizing achieves up to a 51% reduction in Time-to-First-Token (TTFT) and a 53% reduction in KV cache memory footprint with minimal degradation in quality ($\approx -4\%$ Accuracy loss). For video workloads, Scene Change detection for intelligent frame sampling delivers even more substantial gains, achieving up to a 79% reduction in TTFT and a 76% reduction in memory overhead, while retaining competitive task quality. These results establish modality reduction as an essential system-level technique for maximizing the serving capacity and minimizing the latency of MLLMs, providing the foundational data necessary for building next-generation dynamic inference serving systems.

Keywords

MLLM Inference Optimization, KV Cache Management, Model Serving Performance, Modality Reduction, Multimodal Large Language Models

Acknowledgements

The completion of this thesis marks not just an academic milestone, but the culmination of a deeply supportive and collaborative experience. My sincere gratitude goes to the individuals who were instrumental in shaping this work.

Above all, I wish to express my deepest appreciation to my advisor, Dr. Thaleia Dimitra Doudali. Your belief in my abilities and your willingness to offer me the opportunity to conduct my thesis research at the IMDEA Software Institute were the catalysts for this project. Your continuous support and mentorship provided the necessary foundation and resources to explore and complete this work.

I am also profoundly thankful to Konstantinos Papaioannou, the Ph.D. student whose guidance proved invaluable. His patience and expertise during my initial on-boarding process were critical. Konstantinos generously shared his extensive knowledge of the subject matter, helping me navigate complex concepts and effectively structure my research. His dedication to collaboration significantly contributed to the successful formation of this thesis.

Finally, I thank my family and friends for their unwavering patience and support throughout my academic journey.

List of Figures

List of Tables

Contents

1 Introduction

1.1 Motivation

The rapid convergence of computer vision, speech recognition, and natural language processing has culminated in the development of Multimodal Large Language Models (MLLMs) [1], exemplified by systems such as LLaVA and GPT-4V. These models, which leverage the advanced reasoning of Large Language Model (LLM) backbones to process and interpret diverse inputs like images and videos, are quickly transitioning from academic novelty to indispensable components in real-world applications—from sophisticated virtual assistants and complex visual question-answering systems to automated content analysis. However, the deployment and scalability of MLLMs in production environments face a significant, fundamental bottleneck: inference serving performance.

The computational load associated with MLLM inference far exceeds that of text-only LLMs. The high number of parameters, the inherently sequential (autoregressive) nature of token generation, and the necessity of processing large sensory inputs (pixels and frames) simultaneously strain even the most advanced hardware, such as the NVIDIA A100 GPUs used in modern data centers.

1.2 Problem Statement

The challenge of MLLM serving is centered on two interconnected system constraints:

- **Quadratic Complexity in the Prefill Phase:** The Transformer's self-attention mech-

anism requires computation that scales quadratically ($O(N^2)$) with the total sequence length (N). In MLLMs, N includes the text prompt and the embedded visual/video tokens, making the initial prompt processing (Time-to-First-Token, or TTFT) the primary latency bottleneck.

- **KV Cache Memory Exhaustion:** The transient memory state used to store attention Key-Value (KV) pairs, essential for accelerating inference, consumes large amounts of VRAM. The memory footprint of the KV cache grows linearly with the number of modality tokens, directly limiting the maximum number of concurrent requests (throughput) that a serving system, such as vLLM, can handle.

Existing research has successfully tackled these challenges primarily through system-level optimizations for text-only workloads (e.g., vLLM's PagedAttention algorithm for efficient KV cache management). However, few studies rigorously investigate system-level enhancements achieved by intelligently manipulating the complex multimodal input stream itself.

1.3 Research Goal and Hypothesis

This thesis aims to validate and quantify modality reduction as a novel, system-level optimization strategy to boost MLLM inference efficiency. We hypothesize that by reducing the size and quantity of input modalities—while preserving the semantic and structural core of the content—we can achieve substantial gains in performance and memory efficiency with minimal degradation in model quality.

1.4 Key Contributions

Through rigorous experimentation using the high-performance vLLM serving system and a diverse set of MLLM architectures (including dense and sparse MoE models) across three distinct video and three image benchmarks, this work delivers the following key quantitative contributions:

- **Image Optimization Validation:** We demonstrate that proportional image size reduction yields substantial gains in latency, achieving up to a $\mathbf{51}\%$ reduction in

Time-to-First-Token (TTFT) and up to a **53**% reduction in KV cache memory footprint, all while maintaining model accuracy (average loss of only -3.96%).

- **Intelligent Video Optimization:** We identify and validate Scene Change detection as the superior technique for video modality reduction. This intelligent sampling method delivers dramatic efficiency gains, achieving up to a **79**% reduction in TTFT and up to a **76**% reduction in memory overhead, confirming its ability to maximize system capacity while preserving temporal context.
- **Blueprint for Dynamic Serving Systems:** The quantitative trade-offs established between performance gain and quality impact provide the necessary empirical data to inform the development of a next-generation inference serving system capable of on-the-run modality reduction.

1.5 Thesis Structure

The remainder of this thesis is structured as follows:

- **Chapter 2 (Background and Related Work):** Provides the theoretical foundation on the Transformer architecture (including the attention mechanism and KV Cache), the architectural components of Multimodal Models, and the system design principles of LLM Inference Serving Systems, with a focus on vLLM.
- **Chapter 3 (Methodology and Experimental Setup):** Details the experimental components, including the MLLMs used (Qwen2-VL, LLaVA-OV, Pixtral), the video and image datasets utilized, the definition of the two modality reduction techniques, the performance and quality metrics, and the specifications of the inference environment.
- **Chapter 4 (Results and Performance Evaluation):** Presents the empirical results, separated into Image Modality Results and Video Modality Results, quantifying the impact of reduction techniques on latency, memory, and task quality.
- **Chapter 5 (Conclusion and Future Work):** Summarizes the main findings, restates the core contributions, and suggests directions for future research, including the extension of these concepts to audio and multi-modality fusion systems.

2 Background

2.1 Large Language Models (LLMs)

Large Language Models (LLMs) are advanced deep learning models designed to process and generate human-like text. These models are based on neural network architectures and are trained on vast amounts of text data, enabling them to understand context, generate coherent text, and perform a wide range of natural language processing (NLP) tasks.

Architecture

While modern LLMs are predominantly based on the **Transformer architecture** [2] and the **Self-attention mechanism**, historically, other architectures have been used for language modeling, including:

- **Recurrent Neural Networks (RNNs) [3]:** Early language models relied on RNN-based architectures, such as Long Short-Term Memory (LSTM) networks [4], which maintained sequential context but struggled with long-range dependencies.

- **Convolutional Neural Networks (CNNs) [5] for NLP:** Models like ByteNet [6] applied CNNs for language modeling but lacked the flexibility and scalability of Transformers.

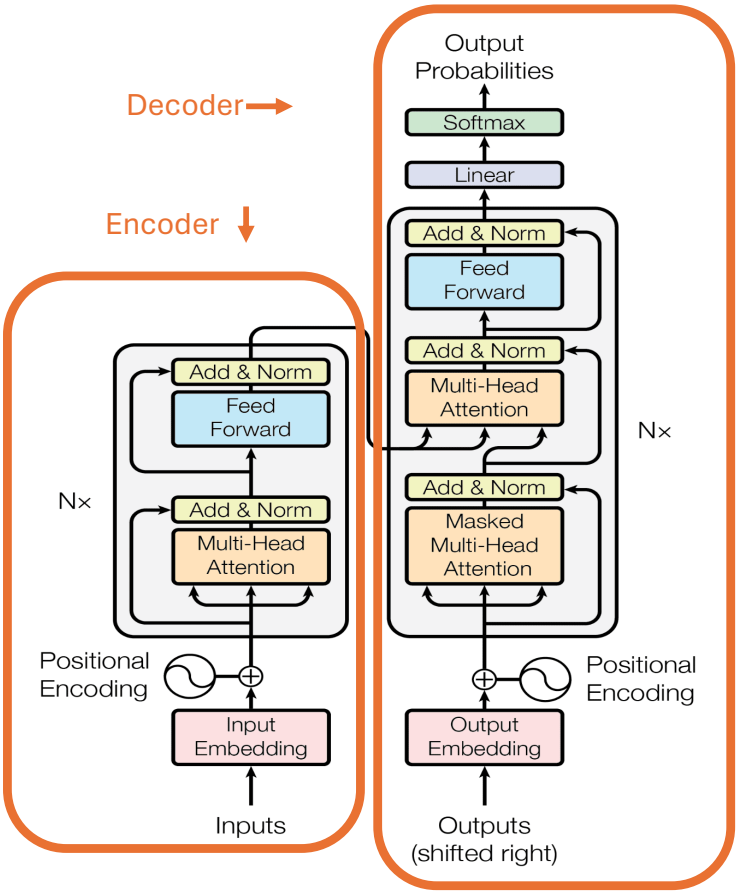


Figure 2.1: *The original Transformer-model architecture as depicted in the paper "Attention Is All You Need"*

Despite these alternative architectures, **Transformer-based LLMs** have become the dominant paradigm due to their superior scalability, efficiency, and ability to capture long-range dependencies using the **Self-attention mechanism**. For the remainder of this section, we focus on Transformer-based LLMs, which form the foundation of state-of-the-art models.

The introduction of the self-attention mechanism has significantly improved the ability of models to capture long-range dependencies in text, leading to breakthroughs in tasks such as machine translation, text summarization, and conversational AI. Modern LLMs, such as GPT-4 [7], LLaMA [8], and Mistral 7B [9], have been further enhanced to follow human-like instructions more effectively.

The architecture of LLMs is primarily based on the Transformer model. We can see a depiction of the Encoder and Decoder Components of the original Transformer Architecture in Image 2.1. The core components of a Transformer-based LLM include:

- **Self-Attention Mechanism:** The attention mechanism allows the model to dynamically weigh different words in a sequence, improving contextual understanding. It enables efficient parallelization and improved handling of long-range dependencies in text. The scaled dot-product attention function is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q, K, and V are the query, key, and value matrices, and d_k is the dimensionality of the key vectors. The softmax function ensures that the attention scores sum to 1.

The matrices Q, K, and V are all linear transformations of the same input embeddings (hence **Self**-Attention) and function like a database look-up:

  - **Query (Q):** Represents the vector of the current word asking for context (i.e., *what to look for*).
  - **Key (K):** Represents the vectors of all words to be matched against the Query (i.e., *what is available*). The product QK^T determines the similarity or relevance score between the current word and all others.
  - **Value (V):** Represents the actual content of all words to be retrieved. These vectors are weighted by the attention scores to produce the final, contextually-rich output for the current word.

A key challenge of the self-attention mechanism is its computational complexity, which scales quadratically with the sequence length, $O(n^2)$. This means that doubling the input length quadruples the memory and computation required. This $O(n^2)$ complexity is the major factor limiting the context window size of LLMs, and it makes the inference process, especially autoregressive decoding, slow without optimizations like the KV-cache (see 2.1).

- **Multi-Head Attention:** Instead of computing a single attention score, multi-head attention allows the model to focus on different aspects of the input sequence simultaneously:

$$\text{MultiHeadAttention}(Q, K, V) = Concat[head_1, ..., head_h]W^0 \tag{2.2}$$

where $head_i = Attention(QW_i^Q, KW_I^K, VW_i^V)$. $Concat$ means to concatenate the attention calculation results of each head, W^0 is the weight matrix of the output later, used to linearly transform the concatenated results [10]. By using multiple attention heads, the model can capture both local and global dependencies, allowing for a more comprehensive understanding of the input sequence. This parallelization also enhances the model's capacity to capture complex relationships between words.

- **Feed-Forward Networks (FFNs):** Each attention layer is followed by a position-wise feed-forward network, improving the model's ability to transform input representations.

- **Positional Encoding:** Since Transformers lack recurrence, they use positional encoding to retain sequential order information. Positional encoding injects information about the position of the words in a sequence. The order of words plays a fundamental part in understanding the semantic meaning of a sentence.

Refinement on Transformer Architectures

While the original Transformer model employed an Encoder-Decoder structure (as shown in Figure 2.1) for sequence-to-sequence tasks like translation, most modern, large-scale LLMs (e.g., GPT, LLaMA, Mistral) adopt a Decoder-Only architecture. This simplified structure exclusively uses the decoder block, which is ideal for causal language modeling (CLM)—the task of predicting the next token in a sequence (see 2.1)—making them highly effective for general-purpose text generation and conversational AI.

Training and Fine-Tuning

In order to enhance their performance on various NLP tasks, LLMs undergo a three-stage training process that includes:

- Data collection and processing.

- Pre-training process that includes determining the model's architecture, pre-training tasks and utilizing parallel training algorithms. Common pretraining objectives include:

  - **Causal Language Modeling (CLM) [11]:** Used in autoregressive models like GPT, where the model learns to predict the next token in a sequence, given the preceding words.
  - **Masked Language Modeling (MLM):** Used in bidirectional models like BERT [12], where certain tokens in a sequence are masked and the model learns to predict them based on the context.

- Fine-tuning (e.g. instruction tuning) and alignment: After pretraining, LLMs can be fine-tuned on specific datasets to adapt them to specialized tasks such as medical diagnosis, legal text analysis, or conversational AI. Recent advancements in LLMs involve instruction tuning, where models are fine-tuned on datasets containing human-written instructions to improve their ability to follow natural language prompts [13]. Additionally, Reinforcement Learning with Human Feedback (RLHF) is used to refine responses based on human preferences, making the model-generated text more aligned with human expectations.

Inference Procedure

LLMs typically follow an autoregressive decoding process during inference, generating one token at a time conditioned on the input prompts and previously generated tokens. The process begins with tokenization, where input text is converted to discrete tokens via subword models (e.g., Byte-Pair Encoding (BPE) [14] or SentencePiece [15]). These are embedded and passed through Transformer layers using self-attention to produce contextualized representations.

The generation process is divided into two major phases:

- **Prefill phase:** It is the first iteration that takes all the input tokens at once and generates the first output token.

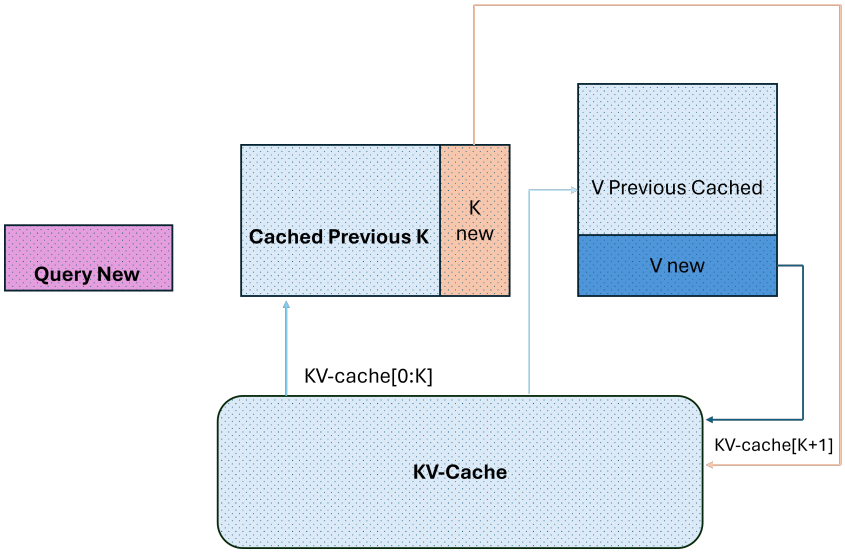


Figure 2.2: *KV Cache: The complexity reduction is achieved because the Key-Value pairs, K and V, from all previously processed tokens (the "Previous Cached" part) are stored in the KV-Cache. For a new token, the model only computes the Query New (Q_{new}), which is then used to compute attention against the existing Cached Keys (K_{prev}) and produce the new context vector by weighting the Cached Values (V_{prev}). This avoids re-running the computation for the entire KV-cache at every step.*

- **Decoding phase:** It handles the generation of subsequent tokens by first taking the output token of the prefill phase and then sequentially creating outputs based on preceding iterations.

To improve inference efficiency, modern systems employ a Key-Value (KV) cache, storing the attention keys and values computed in previous steps. For each subsequent token generation step, only the newly generated token is processed, while previously computed **key-value pairs are retrieved from cache**, eliminating redundant computations and reducing the complexity of each decoding step from **O(n)** (re-computing attention over the entire length n) to **O(1)** (only computing attention for the new token against n cached tokens), where n is the sequence length. The size of KV cache linearly increases with the number of prompt tokens. An illustration of the KV Cache memory is shown in 2.2. A visualization of the Prefill and Decoding Phase of LLM Inference utilizing the KV Cache is shown in 2.3.

At each step, the model samples the next token from the vocabulary distribution using strategies such as greedy decoding or beam search. The process terminates based on pre-defined stopping criteria such as reaching a special end token or maximum length.

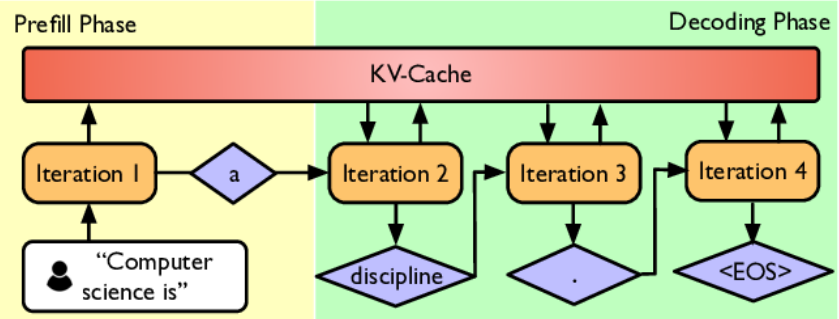


Figure 2.3: *Prefill and Decoding Phase in the LLM inference as depicted in the paper "LLM Inference Serving: Survey of RecentAdvances and Opportunities"*

Applications

LLMs are versatile and power a wide range of applications across industries:

- **Conversational AI:** LLMs serve as the foundation for chatbots and virtual assistants such as ChatGPT, enabling human-like interactions.
- **Content Generation:** These models generate high-quality text for applications like article writing, creative storytelling, and script generation.
- **Code Assistance:** Models like Codex [16] assist in code completion, debugging, and documentation.
- **Translation and Summarization:** LLMs improve machine translation (e.g., Google Translate) and text summarization for research and news articles.
- **Scientific and Legal Document Processing:** LLMs facilitate information extraction, summarization, and analysis of complex documents.

2.2 Multimodal Models

Multimodal models are artificial intelligence (AI) systems that process and combine information from more than one input modality—such as text, images, audio, or video—to improve understanding and decision-making. Unlike unimodal models that operate on a single data type, multimodal systems integrate heterogeneous information sources to achieve more robust and context-aware behavior.

Although multimodal models vary widely in architecture, most share several core components. A typical framework includes modality-specific encoders, a fusion mechanism, and an output decoder:

- **Modality-Specific Encoders:** Each modality is transformed into a vector representation or embedding using a dedicated encoder (e.g., a Vision Transformer for images or a text transformer for language). A multimodal system may include as many encoders as the number of supported modalities.
- **Fusion Mechanism Strategies:** Since different modalities exhibit distinct statistical structures, multimodal fusion aligns their representations into a shared latent space. Common strategies include:
  - **Early Fusion:** Raw inputs from multiple modalities are combined and jointly processed.
  - **Intermediate Fusion:** Each modality is encoded separately and fused in a latent space.
  - **Late Fusion:** Independent modality-specific predictions are merged at the decision stage.
  - **Hybrid Fusion:** Combines multiple fusion strategies across different stages of the model.

  The shared goal is to produce aligned embeddings that allow the model to reason across modalities—for example, determining how well an image corresponds to a caption.
- **Decoder or Output Head:** Depending on the task, the fused representation may be decoded into text, images, class labels, or other structured outputs.
- **Projection/Alignment Layers:** In models composed of independently pre-trained components (e.g., image encoders paired with text-based transformers), lightweight projection layers are often used to map modality-specific embeddings into the dimensional space expected by the core model.

Multimodal models support a broad range of applications, including:

- **Image and Video Understanding:** Captioning, visual question answering (VQA), scene analysis.
- **Cross-Modal Retrieval:** Text-to-image and image-to-text search.
- **Creative AI:** Text-guided image and video generation.

Two landmark multimodal systems are **CLIP** and **DALL·E**. CLIP [17] consists of an image encoder and a text encoder trained using a contrastive objective to learn a shared embedding space, enabling powerful cross-modal alignment. DALL·E is a generative model that creates images conditioned on text prompts; earlier versions used a CLIP-like prior to map textual descriptions into a rich latent representation that guides the image generation process.

While models such as CLIP and DALL·E demonstrated strong modality alignment and generation capabilities, **Multimodal Large Language Models (MLLMs)** extend these ideas by integrating pre-trained Large Language Models with multimodal encoders and projection layers. This design allows an LLM to operate as the central reasoning engine, enabling detailed, context-aware responses to combined visual and textual inputs.

2.3 Multimodal Large Language Models (MLLMs)

Multimodal Large Language Models (MLLMs) are a specialized subset of multimodal models that integrate large language models (LLMs) with visual, auditory, or other sensory data. MLLMs emphasize natural language as a core processing and generation mechanism. These models are designed to understand and generate text-based responses based on multimodal inputs, making them particularly useful for applications such as vision-language understanding, multimodal dialogue systems, and knowledge extraction from diverse data sources, despite computational challenges posed by multimodal data.

MLLMs extend the capabilities of LLMs by incorporating a vision encoder (for images or video) or an audio encoder (for speech or sound) alongside the traditional text-based transformer model. This allows them to perform tasks such as answering questions about images, summarizing video content, or engaging in dialogue grounded in real-world visuals. Some notable MLLMs include LLaVA [18], Flamingo [19], and GPT-4V.

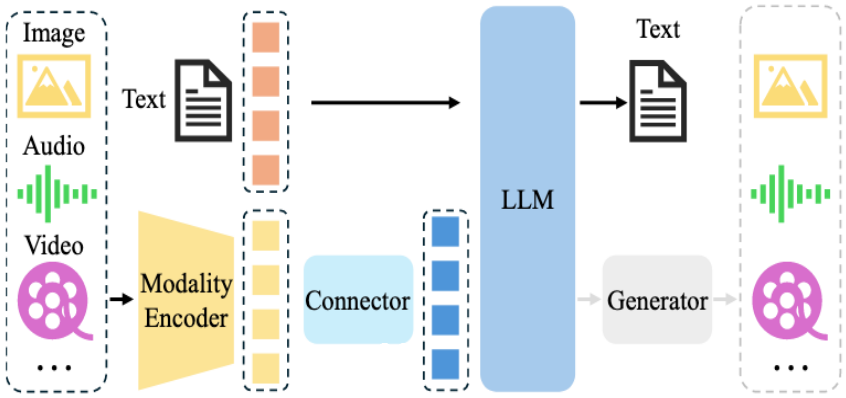


Figure 2.4: *MLLM Architecture*

Architecture of MLLMs

The most common Multimodal Large-Language Model (MLLM) architecture [20] (See Image 2.4) that uses the LLM as a joint part of the system, consists of the following key components:

- **Modality Encoder:** The modality encoder processes raw inputs from a specific modality (e.g., images or audio) and transforms them into dense feature representations. In the case of vision-language models, this is often a Vision Transformer (ViT) [21] or a CNN-based backbone pretrained on large-scale image datasets. These encoders extract meaningful representations from images, videos, or other visual modalities and convert them into feature vectors that can be interpreted by the language model.

- **Language Model Backbone:** The core of an MLLM is a pretrained LLM, such as GPT-4 or LLaMA. The LLM takes the processed multimodal information alongside the sequence of Text tokens (e.g., user prompts or instructions) and generates outputs conditioned on both inputs, performing reasoning, answer generation or captioning using its pre-trained language modeling capabilities, while attending to the non-text inputs. The LLM is typically frozen during training or fine-tuning to preserve its language generation capabilities, while the connector and encoder are fine-tuned to align with it. The output of the LLM is textual and if we don't wish to have any other modality output, this is the last component of our joint model.

- **Connector (Alignment Module):** The connector acts as an interface between the modality encoder and the LLM. Its purpose is to align the modality-specific representations with the token embedding space of the LLM. The output of the connector is a sequence of ”pseudo-tokens” (also called vision tokens or modality tokens) that can be concatenated with textual tokens.

- **Multimodal Generator:** In models designed for multimodal output (e.g., text-to-image), a Multimodal Generator (often a separate decoder or GAN [22]) is used to translate the fused representations into a non-textual output, such as images or video. For most language-centric MLLMs, the LLM backbone serves as the final text generator.

As depicted in Figure 2.4 , the flow begins with Modality Encoders processing different raw inputs (e.g., Image , Text , Audio , Video). The encoded features are then passed through a Connector to align them before being fed into the central LLM backbone. The LLM performs the core reasoning and generates the final Text output via a Generator (or the LLM itself).

Training

A full-fledged MLLM undergoes three stages of training, i.e. pre-training, instruction-tuning, and alignment tuning. Each phase of training requires different types of data and fulfills different objectives:

- **Pre-training:** As the first training stage, pre-training mainly aims to align different modalities and learn multimodal world knowledge. This phase typically uses massive, weakly-curated datasets of image-text pairs or video-text pairs to establish a shared embedding space. Pretraining data mainly serve two purposes, i.e. (1) aligning different modalities and (2) providing world knowledge.

- **Instruction-tuning:** Instruction refers to the description of tasks. Intuitively, instruction tuning aims to teach models to better understand the instructions from users and fulfill the demanded tasks. This involves fine-tuning the model on smaller, high-quality, task-specific datasets that pair a multimodal input with a natural language instruction and a desired output (e.g., ”Describe this image”).

Tuning in this way, LLMs can generalize to unseen tasks by following new instructions, thus boosting zero-shot performance. A multimodal instruction sample often includes an optional instruction and an input-output pair. The instruction is typically a natural language sentence describing the task.

- **Alignment tuning:** Alignment tuning is more often used in scenarios where models need to be aligned with specific human preferences, e.g. response with fewer hallucinations (the phenomenon where the model generates text that is factually incorrect, nonsensical, or not grounded in its training data). Currently, Reinforcement Learning with Human Feedback (RLHF) and Direct Preference Optimization (DPO) are two main techniques for alignment tuning.

  - **Reinforcement Learning with Human Feedback (RLHF):** As examplified in InstructGPT [13], human-annotated feedback is used to optimize multimodal responses further, ensuring factual accuracy and coherence.
  - **DPO [23]:** It learns from human preference labels utilizing a simple binary classification loss.

  The data collection for alignment-tuning is to collect feedback for model responses, i.e. to decide which response is better.

Inference Procedure

The inference process in Multimodal Large Language Models (MLLMs) extends traditional LLM inference by incorporating non-textual modalities such as images, video, and audio. The procedure consists of two main stages:

(1) Encoding the Multimodal Inputs and **(2) Autoregressive Text Generation**.

1. **Input Processing:** Different modalities undergo specific preprocessing steps before being integrated into the model:

   - **Text:** Tokenized and embedded using the same pipeline as LLMs.
   - **Images:** Processed through a **vision encoder** (e.g., CLIP or ViT to extract high-dimensional feature representations.

- **Video:** Individual frames or motion representations are extracted using video encoders (e.g., TimeSformer [24]).
- **Audio:** Converted into spectrogram features, then passed through an audio encoder (e.g., Whisper [25]).

2. **Aligning Modal Representations:** Since LLMs are inherently designed for text processing, non-text embeddings must be projected into the LLM's token space. This is typically achieved using trainable projection layers (e.g., a simple MLP connector) that map visual or audio embeddings to the LLM's embedding dimension.

3. **Integration with LLM Decoding:** Once the multimodal input embeddings are aligned with the LLM's text embeddings, standard LLM autoregressive inference begins. The model generates text token-by-token while attending to both the input text and modality-aligned embeddings:

$$P(t_n|t_1, ..., t_{n-1}, I, V, A) = \text{softmax}(Wh_n) \tag{2.3}$$

where I, V, A represent encoded image, video, and audio features.

4. **Multimodal Attention Mechanism:** During text generation, the LLM's Self-Attention layers function as a Multimodal Attention Mechanism by processing the sequence of text tokens and the aligned modality tokens, allowing the model to selectively focus on relevant visual or auditory details, improving contextual relevance.

5. **Post-Processing:** The final output can take various forms depending on the application:

- **Text Responses:** Multimodal question answering or reasoning.
- **Captioning:** Descriptions of images, videos, or audio.
- **Multimodal Dialogue:** Context-aware chatbot responses that incorporate visual or auditory details.

Applications

MLLMs have transformative applications across various domains:

- **Visual Question Answering (VQA):** Answering natural language queries about images.
- **Image Captioning & Generation:** Generating textual descriptions for images.
- **Multimodal Conversational Agents:** Chatbots that can process images and respond contextually, such as GPT-4V and LLaVA.
- **Document and Chart Understanding:** Processing scanned documents, diagrams, and scientific figures.
- **Video Understanding:** Extracting insights from video content by analyzing both audio and visual components.

2.4 Inference Serving Systems for Large Language Models

Inference Serving Systems [26] are specialized software platforms and optimization engines designed to deploy trained Large Language Models (LLMs) and Multimodal Large Language Models (MLLMs) to production environments. Their core purpose is to maximize throughput (the number of requests processed per second) and minimize latency (the time taken to generate a response), thereby ensuring LLM-powered applications are responsive, cost-effective, and scalable.

The computational and memory demands of modern LLMs [27], which often feature billions of parameters, mean that even high-performance GPU servers can be strained by simultaneous requests and the autoregressive nature of text generation. This challenge necessitates advanced system-level enhancements.

Deploying LLMs at scale presents unique challenges that serving systems must address:

- **High Latency:** Answering natural language queries about images.
- **Memory Constraints (KV Cache):** The Attention Key-Value (KV) Cache, which stores intermediate computation results to avoid redundant calculations during token generation, is a major memory consumer. The size of the KV cache grows linearly with the sequence length and the number of concurrent requests, quickly consuming high-bandwidth GPU memory (VRAM).

This thesis focuses on Serving Systems that apply system-level enhancements—optimizations that maintain the integrity of standard LLM decoding processes—rather than algorithmic modifications to the decoding itself.

The following subsections highlight the key architectural strategies used to overcome these system challenges.

2.4.1 Key System Optimization Techniques

Memory Management and Caching

Efficient memory management is crucial as the KV cache is the most significant transient memory consumer during LLM inference [28]. Recent research focuses on treating the KV cache as a dynamic resource to enable larger batch sizes and longer context processing.

- **Efficient KV Cache Management (PagedAttention):** Recognizing that request generation length and lifetime are not known a priori, systems now manage the KV cache using non-contiguous memory blocks. PagedAttention [29] manages the KV cache akin to operating system virtual memory, significantly reducing memory waste from pre-allocation and fragmentation compared to contiguous KV cache schemes. PagedAttention is now an industry standard, implemented in frameworks like vLLM(which was introduced in the paper that also introduced PagedAttention) and TensorRT-LLM [30].

- **Support for Long-Context Applications:** To overcome memory limits for long sequences, solutions include distributed approaches like Ring Attention [31], which leverages blockwise attention across multiple devices, and Infinite-LLM [32], which breaks the KV cache into small units across distributed memory (rBlocks). Other systems, like InfiniGen [33], offload non-essential KV cache entries to the CPU or prefetch only essential entries to the GPU.

- **Compression of KV Cache:** Due to the large memory footprint, some systems employ compression techniques. FlexGen [34] uses fine-grained groupwise quantization to compress the KV cache to 4 bits. MiniCache [35] leverages the high

similarity between KV cache states in middle-to-deep layers to merge them into a shared representation, reducing redundancy.

Computation Task Scheduling

LLM inference is resource-inefficient on GPUs due to the sequential dependency between tokens. Scheduling optimizations focus on maximizing parallel utilization.

- **Continuous Request Batching:** While simple batching is intuitive, the variable length of responses results in wasted computation when shorter responses wait for longer ones ("tail latency"). Continuous batching, pioneered by Orca [36] and now an industry standard (incorporated into vLLM and TensorRT-LLM) , schedules new requests into the batch as soon as a request in the current batch completes. This maximizes GPU occupancy at the token level.

- **Prefill/Decode Disaggregation:** The prefill and decoding phases have distinct characteristics and resource needs. Disaggregated inference , where the prefill and decode phases run on separate, independently optimized instances, prevents interference between compute-intensive prefill jobs and latency-critical decode tasks.

- **Stall-Free Batching:** Techniques like the dynamic SplitFuse mechanism in DeepSpeedFastGen [37] and Sarathi-Serve's [38] stall-free batching decompose long prompts into smaller chunks and schedule them alongside ongoing decode requests without causing pipeline stalls, thereby maintaining high throughput.

Model Parallelism

For models with hundreds of billions of parameters, distributed execution is mandatory.

- **Heterogeneous Parallelism:** Frameworks like HeteGen [39] employ heterogeneous parallel computing algorithms to distribute computation across a mix of CPUs and GPUs, using asynchronous overlap to mitigate I/O bottlenecks.

- **Sequence Parallelism:** LoongServe [40] introduced Elastic Sequence Parallelism (ESP) to dynamically adapt resource usage between requests and phases (prefilling and decoding), reducing KV cache migration and fragmentation when serving long sequences.

2.4.2 vLLM: High-Performance Inference Engine

The experimental evaluation of MLLM serving performance detailed in this thesis is conducted using vLLM, an open-source, high-throughput, and memory-efficient inference and serving engine for Large Language Models. Originally developed in the Sky Computing Lab at UC Berkeley, vLLM has evolved into a community-driven project with contributions from both academia and industry. vLLM is a fast and easy-to-use library for LLM inference and serving.

The core of vLLM's high performance lies in its architecture, designed to overcome the memory and throughput bottlenecks of autoregressive LLM inference [41].

Architectural Overview

vLLM separates the management of sequence execution and memory into specialized components (See Figure 2.5). The high-level architecture consists of:

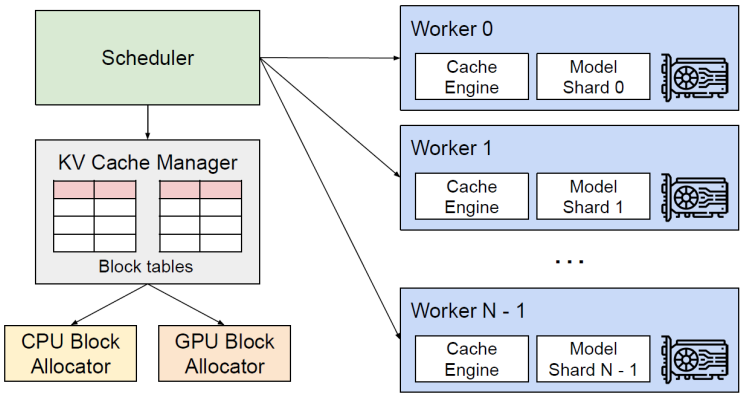


Figure 2.5: *vllm Architecture as depicted in the paper "Efficient Memory Management for Large Language Model Serving with PagedAttention"*

- **Scheduler:** Manages incoming requests and determines which ones should be processed together to maximize GPU utilization.

- **KV Cache Manager:** Oversees the allocation and deallocation of memory blocks for the KV Cache on both CPU (for offloading) and GPU (for active use), utilizing the PagedAttention technique.
- **Workers:** Executes the model computation, with each worker holding a shard of the model and managing its own cache engine for local processing.

PagedAttention: Efficient KV Cache Management

vLLM's primary innovation is the PagedAttention algorithm, which tackles memory fragmentation and waste caused by the dynamic and unpredictable size of the KV Cache.

- **Non-Contiguous Allocation:** PagedAttention manages the KV cache by dividing it into fixed-size blocks (pages), similar to virtual memory in operating systems. This allows the Key and Value vectors for a single request sequence to be stored in non-contiguous blocks of GPU memory.
- **Memory Efficiency:** This block-based approach significantly minimizes memory waste caused by fragmentation and reduces the need for large, over-allocated contiguous memory regions.
- **Block Sharing:** The PagedAttention structure enables key-value block sharing across requests, which is critical for scenarios like long system prompts, multi-turn conversations, or prompt caching.

Continuous Batching for High Throughput

vLLM utilizes continuous batching (also known as iteration-level dynamic batching) to address the underutilization of the GPU that occurs with static batching, which forces shorter sequences to wait for the longest sequence in the batch to complete.

- **Token-Level Scheduling:** Instead of holding completed sequences idle, continuous batching releases GPU resources immediately after a request generates a token or finishes entirely. It then dynamically inserts new requests into the batch queue, ensuring the GPU is continuously processing tasks.

- **Latency vs. Throughput:** This approach primarily optimizes throughput by maximizing GPU occupancy. Since LLM serving workloads are often dominated by decode-intensive operations, continuous batching effectively maintains a high level of parallelism throughout the autoregressive generation process. This contrasts with traditional systems where the GPU utilization would drop off as requests complete.

Multimodal and MoE Support

vLLM is engineered to handle the complexity of modern MLLMs used in this thesis:

- **Multimodal Workloads:** vLLM natively supports Multimodal LLMs (such as the LLaVA family) by efficiently managing the large, fixed-length sequence of visual tokens that must be loaded and cached, which significantly contributes to the overall sequence length and KV cache size.
- **MoE Compatibility:** The engine provides specialized support for sparse models like Pixtral-12B (Mixtral-style MoE), ensuring that only the relevant expert weights are utilized for each forward pass, optimizing execution efficiency.

The combination of PagedAttention and continuous batching makes vLLM the ideal system for conducting a performance analysis where efficiency and throughput are measured under realistic, high-load conditions. For the official vLLM documentation visit [42].

3 Methodology and Experimental Setup

3.1 Multimodal LLMs Under Evaluation

The experimental evaluation of inference serving performance requires a diverse set of Multimodal Large Language Models (MLLMs) to capture the effects of model scale, underlying Large Language Model (LLM) backbone, and vision-language connection methodology. For this study, we selected models encompassing various architectural designs and parameter counts, ranging from ∼0.5 billion to 12 billion parameters.

The models were chosen based on the following criteria:

- **Architectural Diversity:** The selection includes models based on the Qwen2 backbone (Qwen2-VL [43], LLaVA-OV [44]) and a model utilizing a sparse mixture-of-experts architecture (Pixtral-12B [45]), allowing for a comparison of dense versus sparsely activated models. Mixture of experts (MoE) [46] is a machine learning technique where multiple expert networks (learners) are used to divide a problem space into homogeneous regions. MoE represents a form of ensemble learning [47].
- **Scale Variation:** The parameter counts vary significantly (from 0.5B to 12B), enabling a systematic analysis of how model size affects memory footprint, Key-Value (KV) cache utilization, and overall inference throughput.
- **Modality Integration:** Models like LLaVA-OV specifically demonstrate the use of a lightweight connector to align visual tokens with the LLM's embedding space, a critical MLLM design pattern.

Table 3.1: *Overview of Multimodal Large Language Models (MLLMs) used for Performance Analysis.*

Model Name	LLM Backbone	Vision Encoder	Approx. Size (B)	Key Architectural Feature
Qwen2-VL-7B-Instruct	Qwen2 (7B)	ViT	7.5	High-performance Qwen-VL architecture
Qwen2-VL-2B-Instruct	Qwen2 (2B)	ViT	2.5	Smaller-scale Qwen-VL variant
llava_onevision_qwen2_-7b_ov	Qwen2 (7B)	CLIP-ViT-L	7.5	LLaVA-style projector for vision-language alignment
llava_onevision_qwen2_-0.5b_ov	Qwen2 (0.5B)	CLIP-ViT-L	1.5	Smallest model for extreme efficiency analysis
Pixtral-12B	Mixtral-style MoE	ViT (custom)	12 (Activated $\sim$3.5)	Sparse Mixture-of-Experts (MoE) backbone

Detailed Model Descriptions

- **Qwen2-VL Series (`Qwen2-VL-7B-Instruct` and `Qwen2-VL-2B-Instruct`)**
  - **Architecture:** These models are built upon the **Qwen2** large language model, a high-performance decoder-only Transformer. They integrate vision using a dedicated **Vision Transformer (ViT)** encoder to project image features into the language model's input space.
  - **Key Feature:** They represent a general-purpose, deeply integrated MLLM family. The inclusion of the 7B and 2B variants allows for a direct comparison of the impact of the LLM backbone size on inference serving performance, memory footprint, and throughput.
- **LLaVA-OneVision-Qwen2 (`llava_onevision_qwen2_7b_ov` and `llava_onevision_-qwen2_0.5b_ov`)**
  - **Architecture:** These models adhere to the **LLaVA** (Large Language and Vision Assistant) framework. They employ a pre-trained **CLIP Vision Encoder** and a lightweight **Connector/Alignment Module** (e.g., an MLP or Perceiver) to map the visual features into a sequence of "pseudo-tokens" compatible with the Qwen2 LLM backbone.
  - **Key Feature for Optimization:** This architecture isolates the vision-language alignment in the compact Connector, which minimizes changes to the pow-

erful, pre-trained LLM. The crucial design choice is that the number of visual tokens is deliberately fixed and small, regardless of the prompt's length, thus mitigating the quadratic growth ($O(N^2)$) of the KV cache during the prefill phase, directly impacting inference efficiency. The 0.5B variant is essential for analyzing performance at the low-resource end of the MLLM spectrum.

- **Pixtral-12B**
  - **Architecture:** Pixtral is distinguished by its use of the **Sparse Mixture-of-Experts (MoE)** architecture in its language backbone. While the total parameter count is large (12B), only a subset of experts (approximately 3.5B parameters) is activated for any given forward pass.
  - **Key Feature for Optimization:** This model is vital for studying the performance characteristics of sparse MLLMs under a serving load. Its architecture offers a unique trade-off: high memory usage due to all 12B parameters being loaded, yet potentially high throughput due to fewer activated floating-point operations (FLOPs) per token, making it a crucial test case for the vLLM serving framework.

3.2 Datasets and Model Accuracy

The experimental evaluation utilizes a diverse set of standard multimodal benchmarks spanning both image and video modalities, across classification and generative tasks. This provides a comprehensive assessment of the MLLMs' performance and the impact of the serving optimizations on model quality.

3.2.1 Multimodal Benchmarks and Tasks

MMBench_DEV_EN (MMBench Development Set - English)

This benchmark is designed to holistically evaluate the basic capabilities of MLLMs, covering numerous aspects of visual understanding and reasoning.

Table 3.2: *Image Workloads (Image + Text)*

Dataset	Task Type	Key Focus	Output Type
MMBench_-DEV_EN [48]	Multiple Choice Question Answering (MCQA)	General visual reasoning, knowledge, and perception	Classification (A, B, C, or D)
COCO Caption [49]	Image Captioning (Descriptive)	Detailed image description and object recognition	Free-form Text
LLaVABench [50]	Open-Ended Question Answering (Open-QA)	Complex reasoning and detailed textual explanation grounded in the image	Free-form Text

It consists of tasks where the model must select the correct answer from a set of choices (A, B, C, or D) based on a provided image and question. In Figure 3.1 we see the textual prompt that accompanies a sample Image (3.2) from the dataset.

Question:
Which trait does this leaf-cutter ant have?

Hint: This picture shows a leaf-cutter ant. A leaf-cutter ant is a type of insect. Each leaf-cutter ant has a hard outer covering called an exoskeleton. The exoskeleton helps protect the ant's body. This type of ant is called a leaf-cutter because it cuts pieces of leaves off plants. Leaf-cutter ants do not eat the leaf pieces. Instead, they use the pieces to grow their food.

A: It has long, thin legs.
B: The outside of its body is soft.
C: It eats leaves.
Answer: "A"

Figure 3.1: *MMBench_DEV_EN textual sample*

Figure 3.2: *MMBench_DEV_EN image sample*

COCO_VAL (Common Objects in Context - Validation Set)

This is a foundational dataset for image understanding, primarily used to test image captioning capabilities. It requires the model to generate a natural, human-like description of the scene, objects, and their interactions.

Each image in the validation set is typically accompanied by five human-written reference captions (ground truths) which are going to be compared with the result produced by the model. In Figure 3.3 we see the textual prompt that accompanies a sample Image (3.4) from the dataset.

Question:
Please describe this image in general. Directly provide the description, do not include prefix like "This image depicts".

Captions:
'A child holding a flowered umbrella and petting a yak.'
'A young man holding an umbrella next to a herd of cattle.'
'a young boy barefoot holding an umbrella touching the horn of a cow'
'A young boy with an umbrella who is touching the horn of a cow.'
'A boy holding an umbrella while standing next to livestock.'

Figure 3.3: *COCO_VAL textual sample*

Figure 3.4: *COCO_VAL image sample*

LLaVABench (LLaVA-Bench-Wilder)

LLaVABench focuses on open-ended conversation and reasoning, containing challenging, real-world images and complex questions that require deep contextual understanding and detailed, often multi-sentence, answers.

This is an Open-QA task where the model must generate a free-form textual response. In Figure 3.5 we see a textual prompt that contains as Ground-Truth a GPT-4 reply on the question. In Image (3.6) we see the corresponding modal sample.

Question:
What is the name of this famous sight in the photo?

GPT4 Ground-Truth: The famous sight in the photo is Diamond Head.

Figure 3.5: *LLaVABench textual sample*

Video-MME

This benchmark is specifically designed to assess MLLMs' proficiency in video analysis and multi-modal reasoning across diverse, real-world scenarios, covering a broad spectrum of temporal challenges.

It features videos of varying lengths with multiple-choice questions requiring the model to observe, track, and reason over extended time. In Image (3.7) we see a textual example from the benchmark.

Figure 3.6: *LLaVABench image sample*

Table 3.3: *Video Workloads (Video + Text)*

Dataset	Task Type	Key Focus	Output Type
Video-MME [51]	Multiple Choice Question Answering (MCQA)	Temporal reasoning, action recognition, and long-context video analysis	Classification (A, B, C, or D)
MMBench-Video [52]	Open-Ended Question Answering (Open-QA)	Holistic video understanding and free-form temporal reasoning	Free-form Text
TempCompass_Captioning [53]	Video Captioning	Generation of descriptive summaries for complex, temporally-extended video content	Free-form Text

MMBench-Video

This is a rigorous, quantitative benchmark designed to evaluate Large Video-Language Models (LVLMs) in video understanding, utilizing lengthy videos and free-form ques-

Question:
When demonstrating the Germany modern Christmas tree is initially decorated with apples, candles and berries, which kind of the decoration has the largest number?

Video Path: $./video/fFjv93ACGo8.mp4$
Candidates:
A: "Apples" **B** "Candles" **C** "Berries" **D** "The three kinds are of the same number"
Answer: "C"

Figure 3.7: *Video-MME textual sample*

tions. It is highly focused on probing the model's temporal reasoning skills.

It consists of original question-answer pairs over diverse, long-form videos sourced from YouTube. The answers are open-ended, often requiring automated assessment using a high-quality LLM like GPT-4. In Image (3.8) we see a textual example from the benchmark.

Question:
What is the name of the player who scored the first goal in the video?

Video Path: $./video/wZxzBvAgqxc.mp4$
Answer: "Palmer"

Figure 3.8: *Video-MME textual sample*

TempCompass_Captioning

This benchmark, which is part of the broader TempCompass framework, focuses on the model's ability to generate accurate captions or descriptions for video content, emphasizing temporal coherence and understanding dynamic elements.

It's a generative task where the model must produce a summary or description of the video. In Image (3.9) we see a textual example from the benchmark. The "Answer" is used as a general direction for the caption the Models have to make. For example, for the Query provided, Qwen2-VL-7B-Instruct creates the caption: "A 3D digital brain is shown in the video".

Question:
A video and multiple pieces of information will be provided to you. One of these pieces of information matches the content of the video, while the remaining ones do not. Your objective is to pinpoint the information that is in harmony with the video and craft a suitable video caption.

Information A: 'subject': 'entire video', 'direction': 'Zoom out from a 3D digital brain'
Information B: 'subject': 'entire video', 'direction': 'Standing still before a 3D digital brain'
Information C: 'subject': 'entire video', 'direction': 'Zoom into a 3D digital brain' Generated Caption:

Answer: C. Zoom into a 3D digital brain

Figure 3.9: *TempCompass_Captioning textual sample*

3.2.2 Dataset Characterization

To ensure the generalizability of our performance and quality findings, the chosen benchmarks—covering image and video modalities—were characterized by analyzing the distribution of their inherent size properties (pixels for images, frame count for videos). These characteristics dictate the varying computational load and complexity faced by the MLLMs.

Image Dataset Pixel Characterization

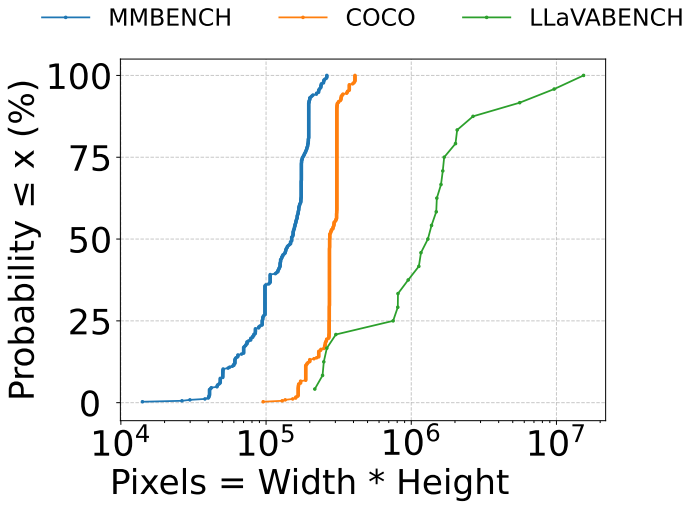


Figure 3.10: *Image Datasets Pixels CDF*

The pixel count (Width × Height) for the image datasets (see Figure 3.10) is highly

diverse, reflecting a wide range of visual complexities encountered in real-world applications .

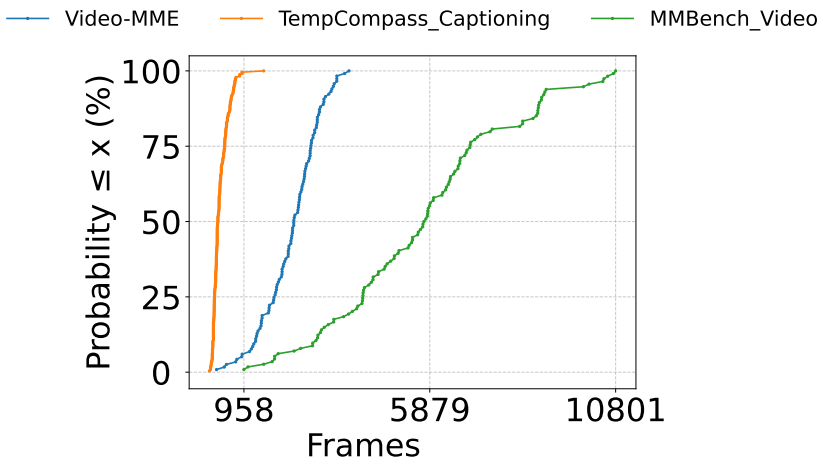
- **LLAVABENCH:** This benchmark exhibits the broadest range of image sizes, spanning from 10^5 up to 10^7 pixels. Its distribution is highly dispersed, with 50% of the images falling below $\approx 3 \times 10^6$ pixels, and the remaining 50% extending to the highest pixel counts. This diversity ensures that the image reduction experiments were tested against inputs ranging from small-scale web images to high-resolution technical diagrams.
- **MMBENCH:** This dataset features images with a generally smaller pixel count. Approximately 90% of its images are clustered below $\approx 3 \times 10^5$ pixels, indicating a focus on small- to medium-sized visual inputs.
- **COCO:** This dataset maintains a tighter distribution than LLAVABENCH, with 90% of its images concentrated between 2×10^5 and 4×10^5 pixels.

The deliberate inclusion of datasets spanning low (MMBENCH) to high (LLAVABENCH) resolution inputs is crucial for validating the robustness of the proportional image downsizing technique across different baseline computational costs.

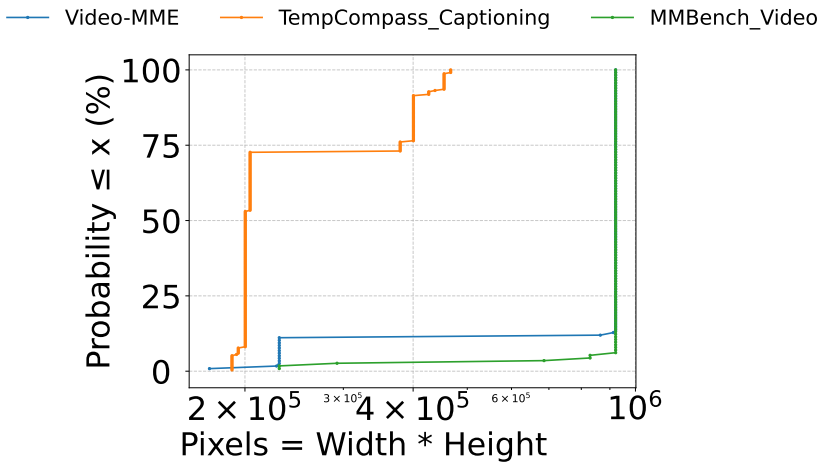
Video Dataset Length Characterization

Video length, measured in total frames (see Figure 3.11), is the primary source of computational and memory heterogeneity in video MLLM serving, as it determines the potential for intelligent frame sampling.

- **MMBench_Video:** This dataset contains the longest videos, with the distribution extending up to $10,801$ frames. The length is highly dispersed, with 50% of the videos containing over $5,879$ frames. This benchmark tests the MLLMs' ability to perform complex, long-context temporal reasoning.
- **TempCompass_Captioning:** This benchmark represents a medium-length video category. Its distribution is tighter, with most videos falling below $\approx 4,000$ frames, making it representative of standard clips used for video captioning.



(a)



(b)

Figure 3.11: *Video Datasets Frames CDF (a) and Video Datasets Pixels CDF (b)*

- **Video-MME:** This dataset focuses on short-duration clips. Almost all ($\approx 99\%$) of the videos are clustered below 958 frames, testing the MLLMs' efficiency in low-latency, short-form video tasks.

While the pixel size of video frames is also diverse, its impact on the LLM bottleneck is secondary to the number of frames sampled. Therefore, the length (frame count) is the key differentiating characteristic for our analysis of temporal reduction techniques.

The experimental evaluation intentionally utilizes datasets that encompass a plethora of sizes and lengths—from short videos and small images to long-context videos and high-resolution images. This heterogeneity in the data ensures that the observed performance benefits (TTFT and memory reduction) and the minimal quality trade-offs

are not artifacts of a narrowly defined input distribution, allowing our results to generalize robustly across a wide range of real-world MLLM applications.

3.2.3 Model Accuracy Evaluation Metrics

To quantitatively assess the trade-off between inference optimization and model quality, two primary metrics are used, chosen based on the task's output structure:

- **Accuracy (for Multiple-Choice Tasks):** Applicable Datasets are MMBench_DEV_-EN and Video-MME. Accuracy is the simplest and most direct metric for classification tasks. It is calculated as the ratio of correctly answered questions to the total number of questions.

  $$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}}$$

  Since these benchmarks have discrete, single-token correct answers (A, B, C, or D), Accuracy provides a definitive measure of the model's categorical reasoning ability.

- **ROUGE-L (for Generative Tasks):** Applicable Datasets are COCO_VAL, LLaVABench, MMBench-Video, and TempCompass_Captioning.

  ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of automatic evaluation metrics commonly used for summarization and text generation. ROUGE-L is a specific variant based on the Longest Common Subsequence (LCS) between the model-generated output and a human-written reference. The LCS represents the longest sequence of words that appears in both texts in the same order, although not necessarily contiguously.

  $$\text{Rouge-L} = \frac{\text{LCS}(\text{Prediction}, \text{Reference})}{\text{Length}(\text{Reference})}$$

  ROUGE-L is used in multimodal generative tasks—such as image captioning or open-ended visual question answering—because it preserves word order and captures sentence-level fluency better than simple n-gram–based metrics. By measuring the longest shared sequence, ROUGE-L rewards models that generate text with similar structure and content to the reference.

However, ROUGE-L has important limitations. It does not measure deep semantic similarity, contextual correctness, or reasoning quality. Paraphrased but semantically equivalent outputs may receive a low ROUGE-L score simply because they use different wording. Despite these shortcomings, ROUGE-L remains widely used in benchmarking because it is simple, reproducible, language-agnostic, and correlates reasonably well with human judgments in many captioning-style tasks.

3.3 Serving System Performance and Efficiency Metrics

The evaluation focuses on measuring both the raw inference performance (speed and throughput) of the MLLMs under load and the direct impact of architectural elements on memory efficiency (VRAM usage).

Inference Latency and Speed

In LLM serving, the overall time to complete a request (end-to-end latency) is composed of two distinct parts: the prefill phase and the decode phase. See 3.4.

The TTFT is emphasized in this analysis, especially because several tasks, such as Multiple Choice Question Answering (MMBench, Video-MME), often require the model to generate only a single classification token (A, B, C, or D) or a very short answer. In these scenarios, the prefill time (TTFT) constitutes almost the entire latency of the request.

Memory and Architectural Efficiency

To quantify the memory footprint of the MLLM workloads, we measure the factors that determine the size of the KV Cache, the primary memory consumer during inference. See 3.5.

Table 3.4: *Inference Latency Metrics*

Metric	Definition	Rationale for Use
Time-to-First-Token (TTFT)	The duration from when the request is sent to the server until the first output token is generated.	TTFT directly measures the efficiency of the prefill phase, which is the computational bottleneck dominated by $O(N^2)$ attention and the processing of the entire input, including the multimodal features. This is the critical metric for interactive performance.
Time-per-Output-Token (TPOT)	The average time taken to generate each subsequent token after the first one.	TPOT measures the speed of the decode phase, which is typically memory-bound and directly influenced by the efficiency of KV cache access and management.
Throughput (Tokens/s)	The total number of output tokens processed by the system per second.	Measures the system's overall capacity to handle parallel requests, reflecting the efficiency of the serving system (e.g., vLLM's continuous batching).

3.4 Experimental Hardware Setup

The performance analysis was conducted on a dedicated server environment configured to support high-throughput, memory-intensive LLM and MLLM inference workloads. This controlled environment ensures that reported metrics accurately reflect system-level efficiency gains rather than external resource contention. See Table 3.6.

This setup is representative of a typical high-performance enterprise server configuration used for deploying large foundation models, allowing for the isolation and measurement of performance benefits derived specifically from the system-level optimizations studied.

Table 3.5: *Memory Footprint Metrics*

Metric	Definition	Rationale for Use
Modality Token Count ($N_{modality}$)	The fixed number of visual or video tokens generated by the encoder and passed to the LLM per multimodal input instance.	This is the direct factor influencing the KV cache size, as the memory requirement is $N_{total} \times BatchSize$, where N_{total}=N_{text}+$N_{modality}$. Analysis of this parameter links the architectural design of the MLLM directly to inference resource usage.
KV Cache Memory Footprint	The total VRAM occupied by the attention Key-Value pairs across all concurrent sequences.	This measured metric reflects the effectiveness of vLLM's PagedAttention and provides a quantitative measure of the memory required for serving, directly limiting the system's maximum batch size.

3.5 Modality Input Size Reduction Techniques

The core of this experimental methodology is to analyze the trade-off between inference serving performance (measured by metrics like TTFT and throughput) and model quality (measured by Accuracy and ROUGE-L) when the multimodal input data is strategically reduced. This is achieved by comparing the performance using original inputs against the same queries using reduced modality inputs.

3.5.1 Image Modality Reduction (Size Downsizing)

For image workloads, the primary reduction technique focuses on the physical size of the image input.

- **Technique:** We reduce the image size proportionally across both width and height, testing various reduction ratios (e.g., $5\%, 10\%, \ldots$, up to 90% of the original size).
- **Rationale:** This approach intentionally maintains the original aspect ratio of the image. Maintaining the aspect ratio is crucial because it ensures the relative position of elements and objects within the scene remains geometrically accurate.

Table 3.6: *Inference Environment Details*

Component	Specification	Notes
GPU	NVIDIA A100-PCIE-40GB	The primary computational accelerator, providing 40GB of High Bandwidth Memory (HBM) crucial for holding model weights and the large KV cache generated by the MLLMs.
CPU	2.3 GHz Dual-Core Intel Core i5	Used for managing the operating system, scheduling, and pre- and post-processing tasks; inference computation is primarily GPU-bound.
Operating System	Linux (Debian 6.1.0-40-amd64)	Stable Linux kernel environment for reliable deep learning operations.
CUDA Driver/Toolkit	CUDA V12.4.131	Provides the necessary libraries and runtime for optimizing GPU kernel execution, supporting high-speed operations within the serving framework.
Serving System	vLLM 0.8.4	The core inference engine used for all experiments, leveraging PagedAttention and continuous batching to maximize GPU utilization.

While other non-proportional reduction techniques were explored, they often distort spatial relationships, leading to a significant and often unpredictable drop in the MLLM's reasoning and captioning quality. By preserving the aspect ratio, we aim to maintain model quality despite reducing the computational load.

Another idea tested was converting the images to grayscale. This though didn't give back any improvement, because on the one hand the number of created tokens only depends on the number of image pixels and not their content and on the other hand by changing the colors, some image details are hidden from the models, so their efficiency gets worse.

3.5.2 Video Modality Reduction (Intelligent Frame Sampling)

Since MLLMs cannot process every frame of a video, they rely on a fixed, user-defined maximum number of frames (N_{frames}, typically 4, 8, 16, 32, or 64). The default sampling technique is uniform, which always selects N_{frames} evenly across the video duration.

Our approach is to replace this brute-force uniform method with intelligent, content-aware sampling techniques that select fewer frames ($N_{\text{frames}} \ll N_{\text{max}}$) while retaining key information, thus improving memory efficiency without a proportional loss in quality.

- **Scene Change Detection:** This technique prioritizes frames that mark a significant visual transition, assuming that these scene cuts encapsulate the start and end of distinct events or contexts within the video.
  - Concept: Identify moments where the visual content changes significantly (scene cuts) using the ContentDetector [54] algorithm.
  - Mechanism: The detector compares successive frames based on the weighted average of HSV (Hue, Saturation, Value) change. If the difference is above a defined threshold, a scene cut is registered.
  - Frame Selection: The function selects the first, middle, and last frames of every detected scene to summarize its content. If too many frames are selected, they are uniformly thinned down to the `max_frames` cap. If no scenes are found, it falls back to uniform sampling.
- **Motion-Based Sampling:** This technique prioritizes moments of activity, assuming frames showing significant motion carry more information about dynamic actions and events.
  - Concept: Prioritize frames where significant motion or relevant activity occurs, as these frames often contain the most important information regarding actions and events.
  - Mechanism: The function calculates the dense optical flow between consecutive frames using the Farneback method [55]. The optical flow measures pixel movement.

  - Frame Selection: Frames where the average motion magnitude across the frame exceeds a set `motion_threshold` are flagged as 'active' and selected. This ensures that the model focuses on segments of the video where relevant action is taking place.

- **Sharpness-Based Sampling:** This technique focuses on eliminating blurry or low-quality frames that may introduce noise or unnecessary artifacts, ensuring the frames passed to the MLLM are visually reliable.

  - Concept: Discard blurry or low-quality frames and prioritize sharper frames, as high visual fidelity is crucial for tasks like Object Recognition and VQA.
  - Mechanism: The function measures frame sharpness using the variance of the Laplacian operator. The Laplacian is highly sensitive to edges and high-frequency content.
  - Frame Selection: Frames with a sharpness score (Laplacian variance) above the `sharpness_threshold` are selected. This acts as an initial quality filter, ensuring that the visual tokens derived from these frames are based on clear, useful image data.

Video Comparative Analysis of Video Frame Reduction Techniques

To determine the most efficient and robust technique for video modality reduction, we compared the four sampling strategies—Uniform (baseline), Motion-based, Sharpness-based, and Scene Change-based—across all five MLLM architectures and three video benchmarks (Video-MME, MMBench-Video-QNA, and TempCompass-Captioning). The analysis focuses on the core trade-offs: minimizing memory footprint and latency while maintaining model quality.

- **Memory Footprint (Modality Token Count):** The memory footprint, directly related to the size of the KV cache, is proportional to the number of modality tokens (N_{modality}) passed to the LLM. As illustrated in the Memory Footprint plot (Figure 3.14):

  - Uniform Sampling consistently results in the highest memory footprint across all models and tasks. This is because uniform sampling always selects the

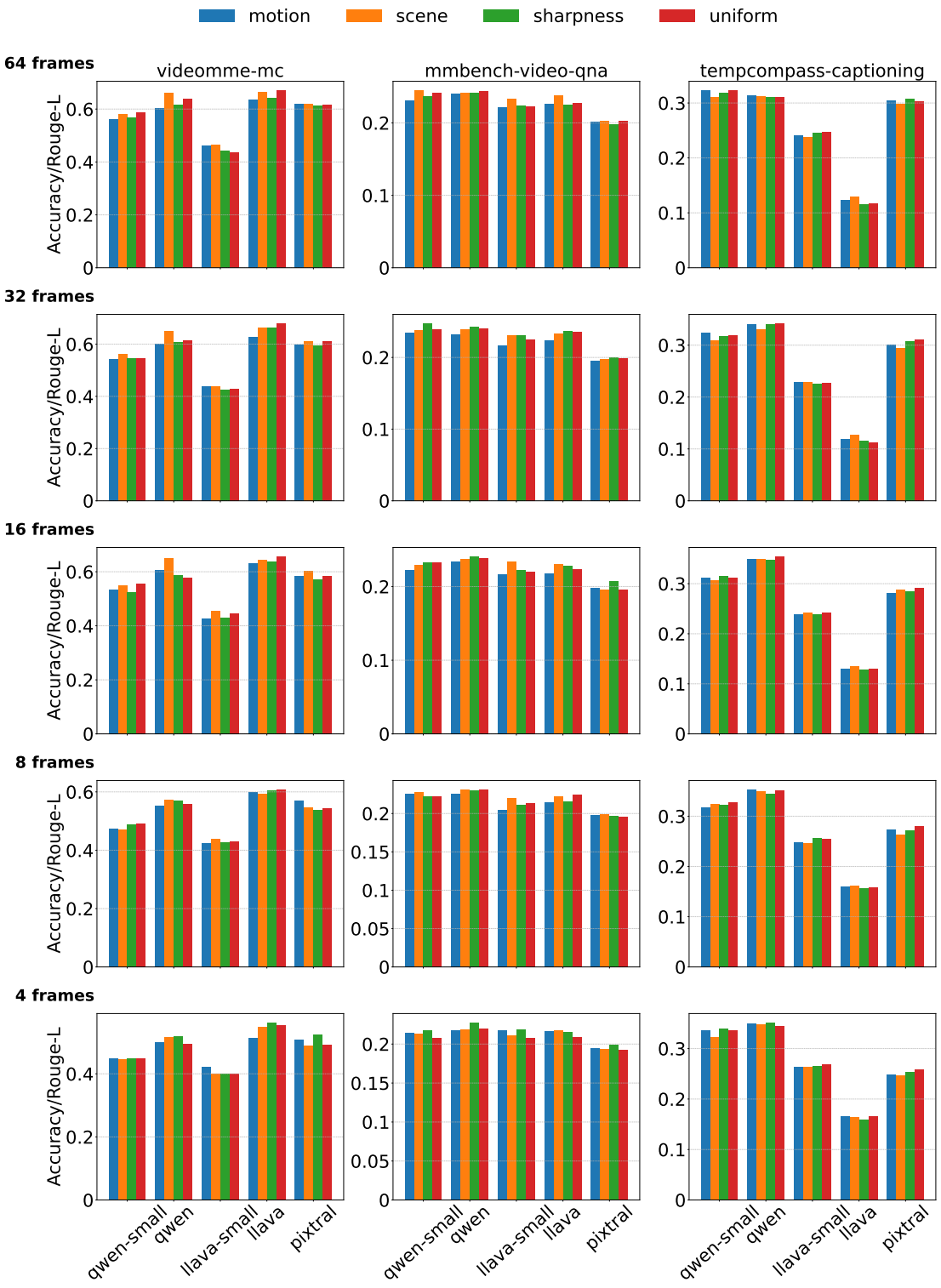


Figure 3.12: *Scores of all models and frame sampling techniques on all the datasets*

maximum allowable number of frames (N_{max}, e.g., 64 frames), regardless of the video content, resulting in the maximum possible N_{modality}.

– Motion, Sharpness and especially Scene Change sampling successfully re-

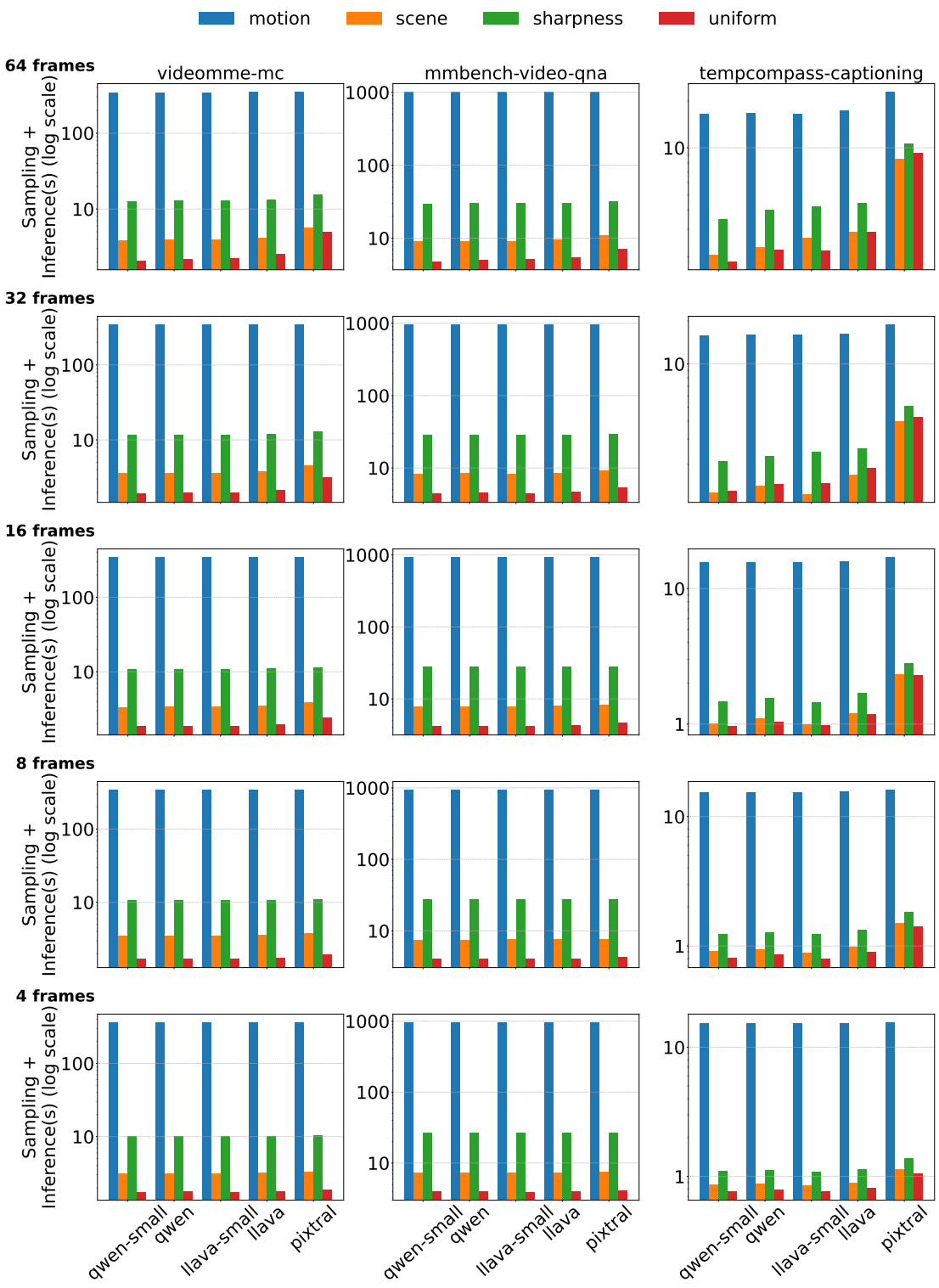


Figure 3.13: *Inference Latency + Frame Sampling Latency of all models and frame sampling techniques on all the datasets*

duce the memory footprint compared to Uniform sampling, particularly for larger $N_{\max}$ caps (32 and 64 frames). This reduction occurs because these

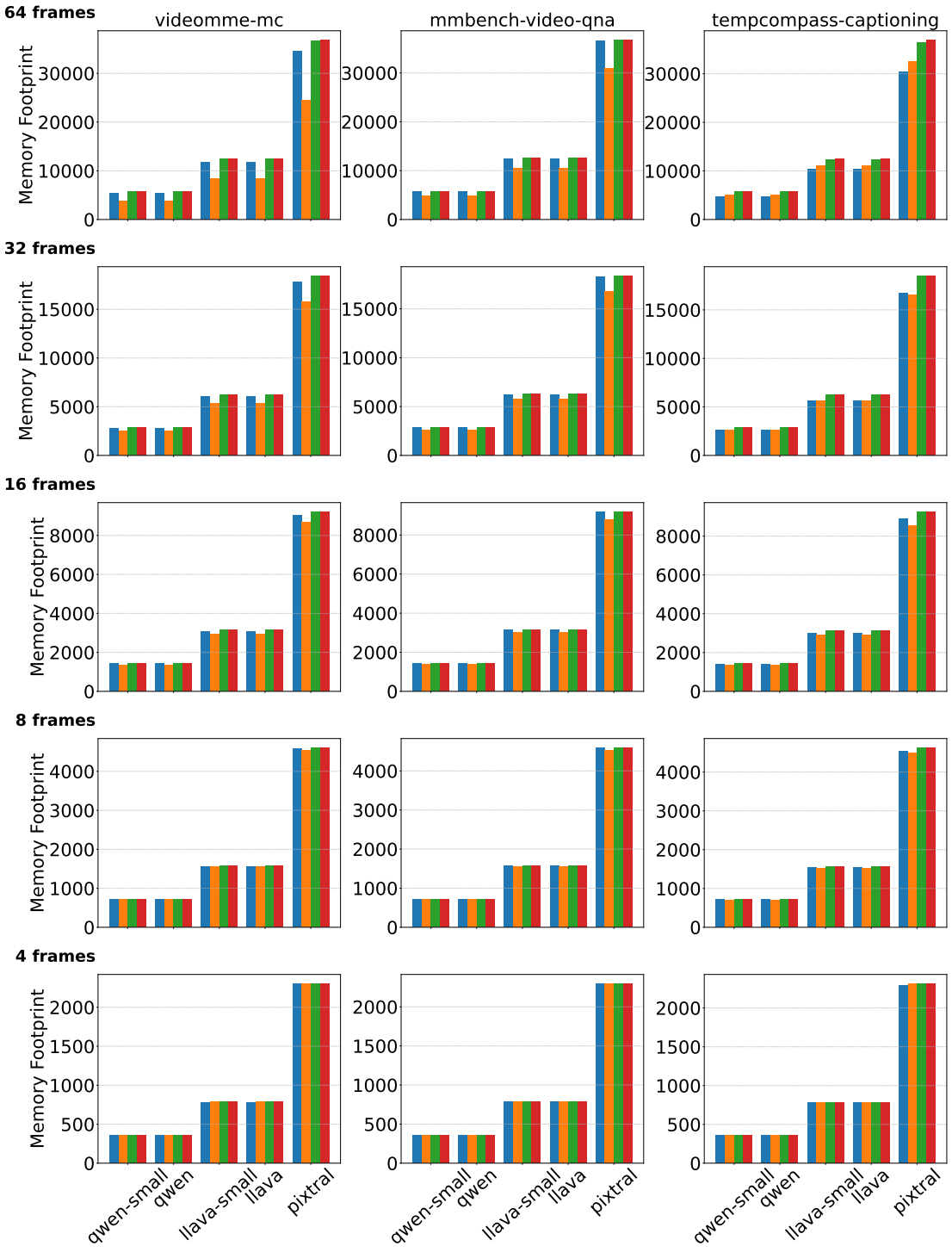


Figure 3.14: *Modality Footprint (Number of Modality Tokens) of all models and frame sampling techniques on all the datasets*

techniques select a number of frames $N_{\text{selected}} \leq N_{\text{max}}$.

– The reduction advantage is most pronounced for large N_{max} values (e.g., 64 frames). This confirms the initial hypothesis: intelligent sampling is most

beneficial in scenarios where reducing the token count provides the maximum relief to the $O(N^2)$ memory and computational cost.

- **Performance (Inference Time + Sampling Latency):** The total performance is measured by combining the preprocessing time (frame sampling) with the inference time (TTFT + decode). As illustrated in the Frame Sampling + Inference Ltency plot (Figure 3.13):
  - The Motion-based technique introduces enormous preprocessing latency in CPU, seriously affecting the whole inference procedure.
  - Among Sharpness-based and Scene-Change techniques, the second is better introducing less preprocessing latency in CPU time. Even though, the total depicted latency for Scene-Change is bigger than the latency of the Uniform technique, this is not necessarily a problem. Frame Sampling Preprocessing introduces CPU latency but if there are token gains (gains in how much the KV Cache is affected), the later latency in the GPU will be less. In general, this is a trade-off companies and users usually prefer.
- **Model Quality (Accuracy/ROUGE-L):** The Quality plot assesses the essential trade-off: whether memory savings compromise model utility. As illustrated in the Scores plot (Figure 3.12):
  - The Scene Change method consistently achieves quality metrics (Accuracy/ROUGE-L) that are competitive with, and in several instances, superior to, the Uniform baseline across all models and tasks. For example, in Video-MME-MC and MMBench-Video-QNA, Scene Change often slightly outperforms the Uniform method.
  - The Sharpness and Motion methods generally perform well but show slightly more variability or marginal degradation compared to Scene Change.
  - The superior performance of the Scene Change technique suggests that identifying distinct events (scenes) is a highly effective heuristic for retaining the semantic core of a video, often providing better temporal context than a simple uniform distribution of frames.

Based on the multi-metric analysis, the Scene Change-based sampling technique is the superior choice for subsequent thorough experiments:

Memory Efficiency: It successfully reduces the memory footprint and modality token count, especially for large N_{max} scenarios, directly addressing the KV cache size challenge.

Quality Retention: It achieves a similar or often better quality score than the Uniform (maximum frame) baseline. This is the crucial finding: system efficiency is improved without compromising task accuracy.

Therefore, the Scene Change-based reduction technique will be used to explore the full range of N_{max} settings in the main performance analysis, as it offers the best balance between system resource optimization and MLLM task performance.

4 Results and Performance Evaluation

This chapter presents the empirical results of the performance analysis, evaluating the impact of modality reduction techniques on both the inference serving performance (latency, throughput, and memory footprint) and the task quality (Accuracy and ROUGE-L) of the selected Multimodal Large Language Models (MLLMs). The analysis is divided into two primary sections, addressing the distinct optimization strategies applied to image and video modalities.

4.1 Image Modality Results

The experiments on image workloads focus on quantifying the trade-off between reducing image size—a technique that minimizes the $O(N^2)$ prefill cost by reducing the resultant modality token count ($N_{modality}$)—and maintaining task performance on Vision Question Answering (VQA) and Image Captioning benchmarks.

4.1.1 Baseline Latency Analysis

To establish a baseline for potential optimization, we first analyze the component latency breakdown for all five MLLM architectures when processing high-resolution original image inputs across the three image benchmarks: LLaVABench-QNA, COCO-VAL Captioning, and MMBench-MC.

As illustrated in the baseline latency plot 4.1, a significant portion of the total latency is attributed to the initial preprocessing and encoding stages of the multimodal input

before the tokens reach the Large Language Model (LLM) backbone.

- **Preprocess and Encoder Time:** For nearly all models, the combined time spent on Preprocess (converting the raw image to features) and Encoder (transforming features into $N_{modality}$ tokens) represents an important percentage of the total latency. In some cases, such as the LLaVABench task for the `llava-small` and model, the preprocess time alone consumes a substantial fraction of the overall time-to-first-token (TTFT). This overhead is due to the inherent complexity and sequential nature of passing large image inputs through the Vision Encoder (e.g., CLIP-ViT-L).

- **LLM Dominance (Total Latency):** While the encoder time is high, the total latency is still dominated by the LLM (Decoder) computation time, especially for models with large backbones like `llava` (7B) and the sparse `pixtral`.

The high cost of the initial encoding phase suggests a clear avenue for optimization. By reducing the size of the input image, we aim to reduce the complexity and execution time of the Encoder. We hypothesize that strategic downsizing can achieve the following simultaneous gains:

Improve Memory Footprint: A smaller image input ultimately contributes fewer or less computationally-intensive visual features, indirectly reducing the computational overhead related to the KV cache, thereby improving overall system throughput.

- **Reduce Encoder and Preprocess Latency:** By processing fewer pixels, the time spent in the initial stages should decrease.

- **Maintain Task Quality:** By ensuring the original image aspect ratio is maintained, we seek to preserve the relative spatial information critical for reasoning and captioning, thus minimizing the drop in Accuracy and ROUGE-L scores.

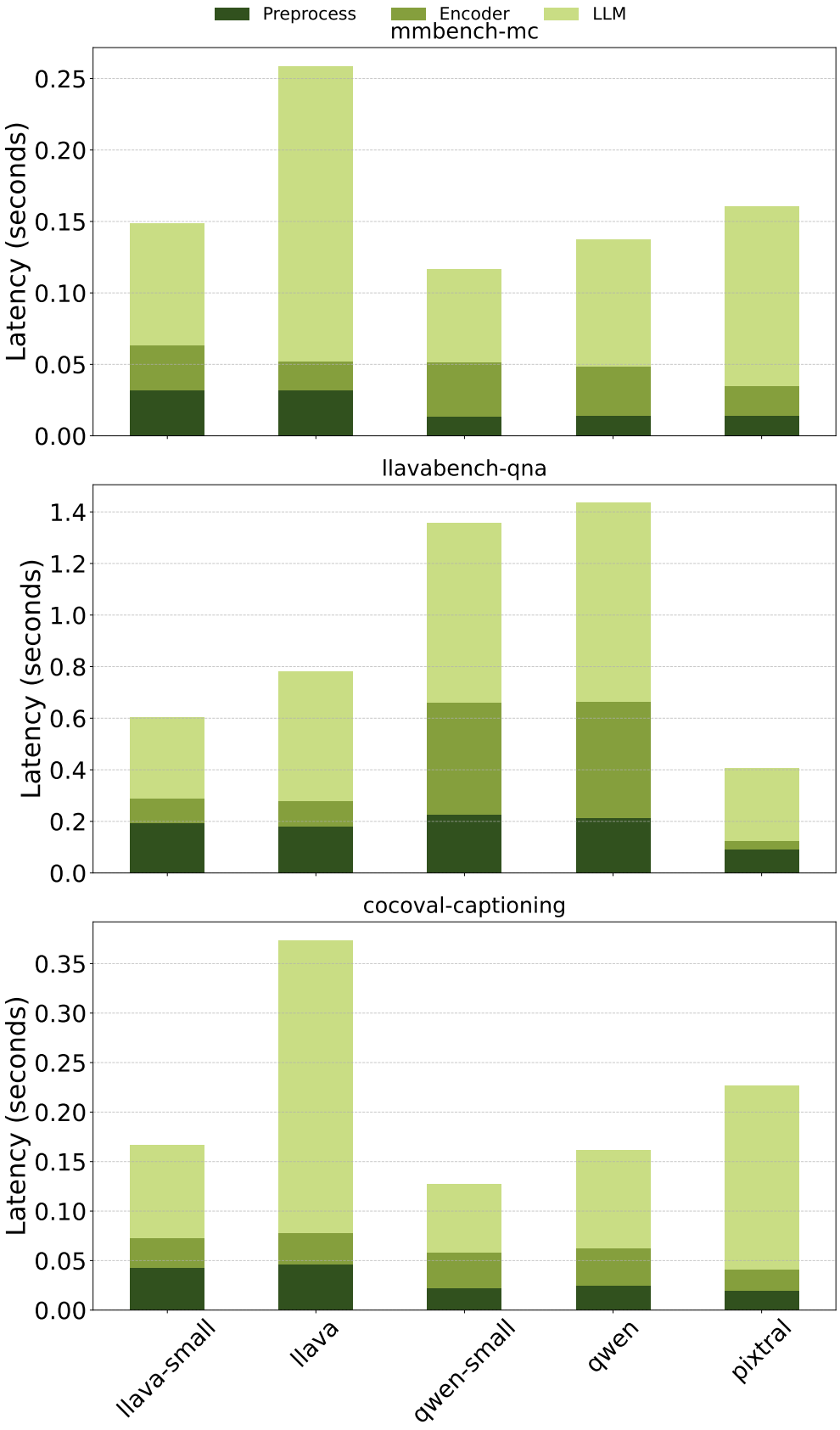


Figure 4.1: *Latency Breakdown for All Image Tasks across all Models*

4.1.2 Modality Reduction Results

We compare the results using the original image size (0% reduction) against proportional downsizing up to 90%.

The quality metrics (Accuracy for MMBench-MC and ROUGE-L for generative tasks) reveal how robust the models are to significant image size reduction (see Figure 4.2).

- **Overall Accuracy:** For the MMBench Multiple-Choice (MC) task, the model Accuracy tends to decrease by only a small margin, averaging -3.96% across all models and reduction techniques. This minimal degradation is highly favorable, considering that the images used in some scenarios (70%, 80%, 90% reduction) are dramatically smaller than their originals. This supports the initial hypothesis that maintaining the aspect ratio preserves the essential spatial context necessary for general visual reasoning.

- **Generative Tasks (ROUGE-L):** The results for open-ended tasks show greater fluctuation:

  - COCOVAL-Captioning: The average ROUGE-L score surprisingly increases by 2.712% across all models and reduction levels. This fluctuation is likely due to ROUGE-L's sensitivity to small changes in short, generated responses.

  - LLaVABench-QNA: This task is more sensitive to image size, showing an average ROUGE-L decrease of -4.54%. The complex, open-ended nature of the questions in LLaVABench likely demands finer visual details, making it more dependent on the original image size.

- **Conclusion on Quality:** Overall, the quality scores demonstrate that the models are highly resilient to image downsizing. The minimal reduction in accuracy across all scenarios is a positive finding, validating the method of proportional downsizing as a viable system optimization.

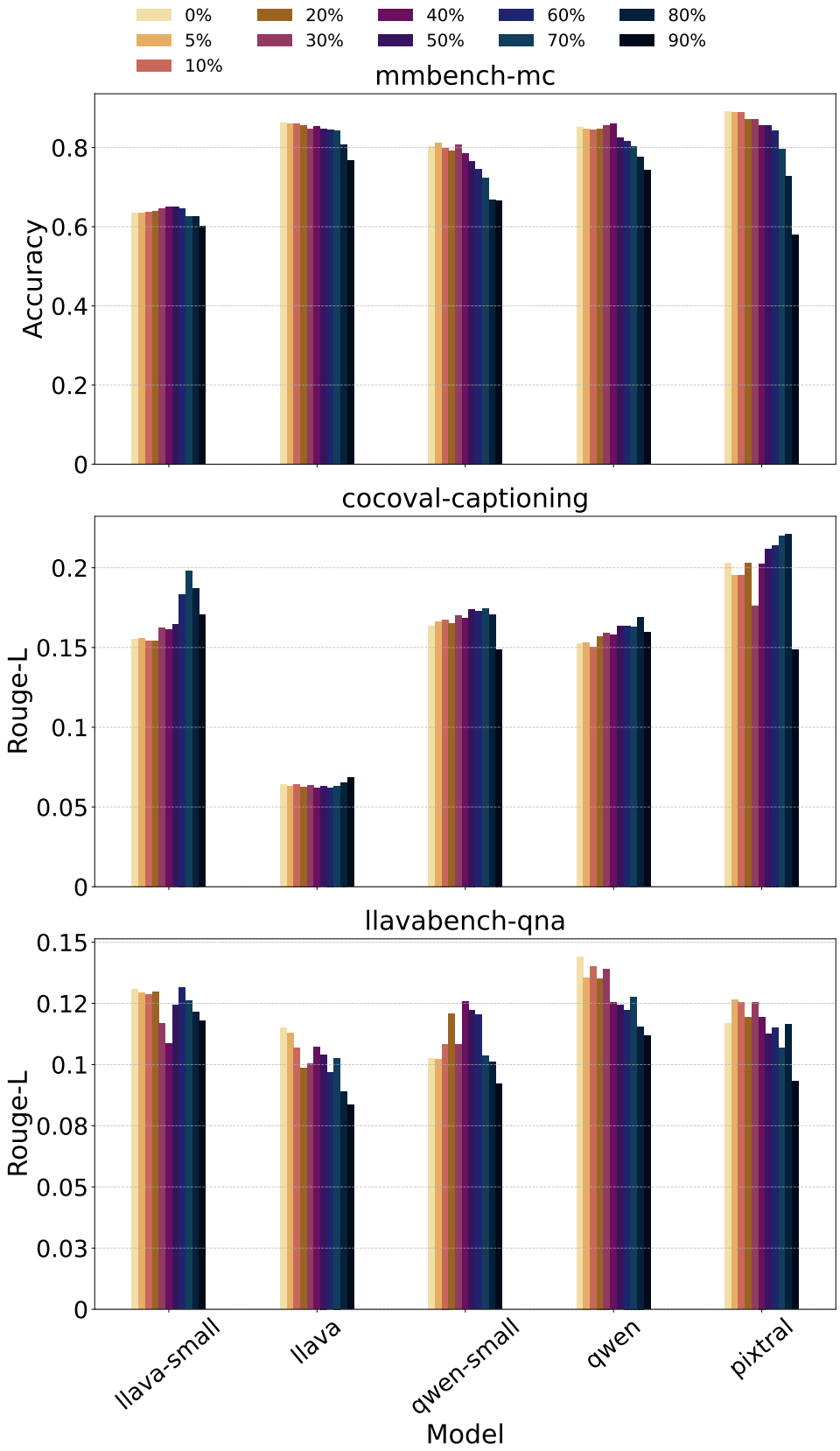


Figure 4.2: *Accuracy and Rouge-L Scores for All Image Tasks across all Models and all Percentage Reductions*

The relative latency plot visualizes the reduction in Time-to-First-Token (TTFT) when processing smaller images, relative to the 0% reduction baseline . Since TTFT is dominated by the $O(N^2)$ prefill computation, its reduction directly measures the benefit of the optimization (see Figure 4.3).

- **Average Latency Gains:** The reduction in latency is significant across all tasks:
  - MMBench-MC: Average relative reduction across all models and reduction percentages is 0.77x (or a 23% speedup).
  - COCOVAL-Captioning: Average relative reduction is 0.63x (or a 37% speedup).
  - LLaVABench-QNA: Average relative reduction is the most dramatic at 0.49x (or a 51% speedup).
- **Task Dependence:** The difference in latency reduction across tasks (MMBench vs. LLaVABench) is likely due to the prefill length heterogeneity of the datasets. LLaVABench's complex prompts and questions likely have longer text inputs, meaning the reduction in image-derived tokens has a proportionally larger impact on the total sequence length N, thereby maximizing the benefit of the $O(N^2)$ reduction.
- **Model Specifics:** Models such as `qwen-small` and `pixtral` show the steepest drop in latency, with `qwen-small` reaching nearly 0.1x its original latency in LLaVABench-QNA at 90% reduction. This highlights that certain model backbones benefit more from input slimming.

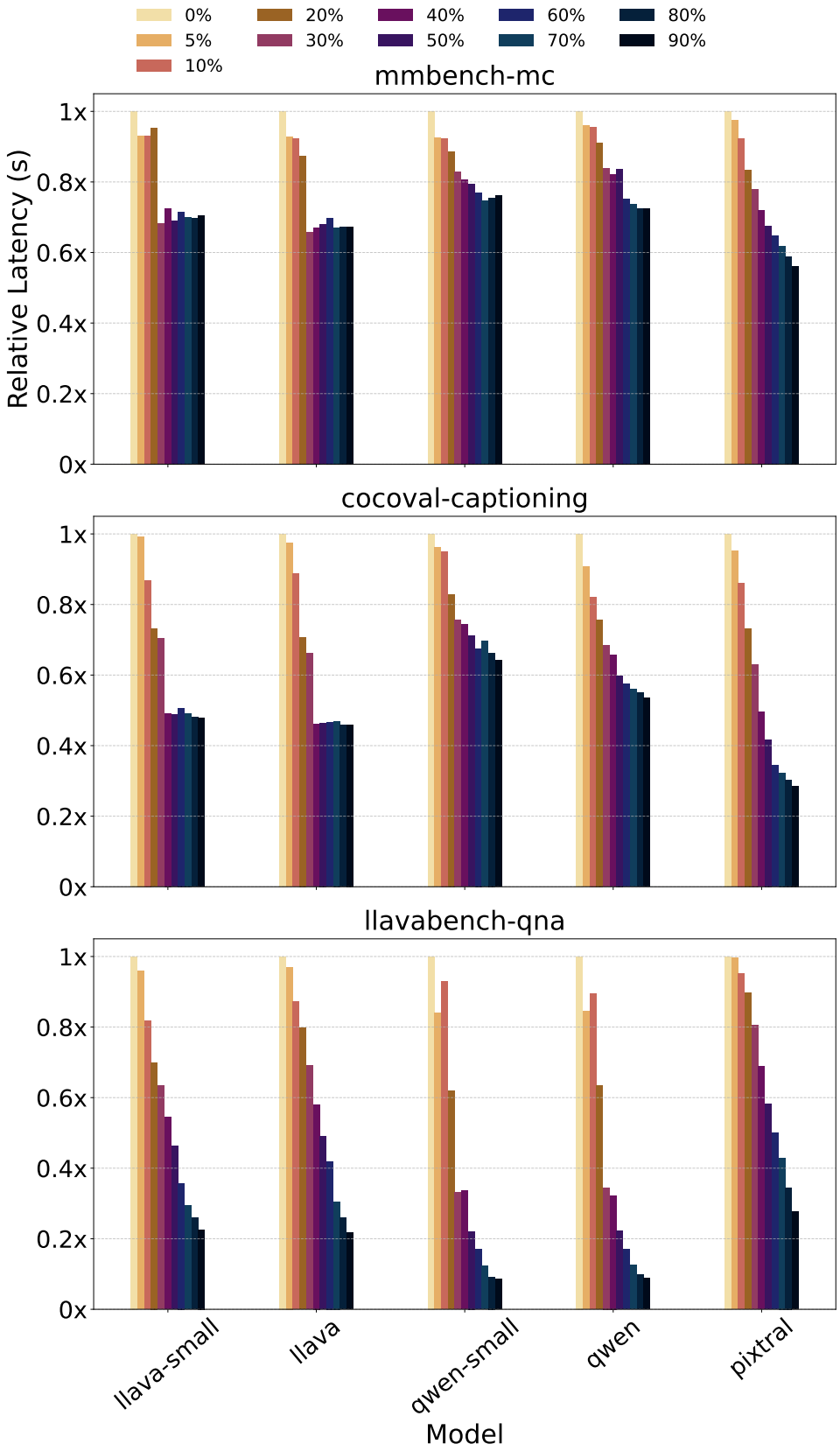


Figure 4.3: *Relative Latency Reduction for All Image Tasks across all Models and all Percentage Reductions*

The relative memory footprint plot shows the reduction in the total VRAM required to store the KV cache, relative to the 0% reduction baseline . Since the memory footprint scales with the number of tokens, this reduction directly confirms the architectural benefit of image downsizing (see Figure 4.4).

- **Average Memory Reduction:** The reduction is consistent and substantial across all tasks, confirming the efficiency of the technique:
  - MMBench-MC: Average relative memory reduction is 0.53x.
  - COCOVAL-Captioning: Average relative memory reduction is 0.47x.
  - LLaVABench-QNA: Average relative memory reduction is 0.51x.
- **Correlation with Latency:** The observed memory reduction closely tracks the latency reduction, validating the connection between modality input size, memory overhead, and computational cost. By reducing the number of input tokens derived from the image, we reduce the total number of KV pairs that must be allocated and managed by vLLM's PagedAttention, allowing for a greater effective batch size and higher throughput.

In summary, proportional image downsizing provides substantial gains in both latency and memory efficiency (up to 51% reduction in TTFT and 53% reduction in memory) while introducing negligible performance degradation for most tasks, making it a powerful system-level optimization for MLLM serving.

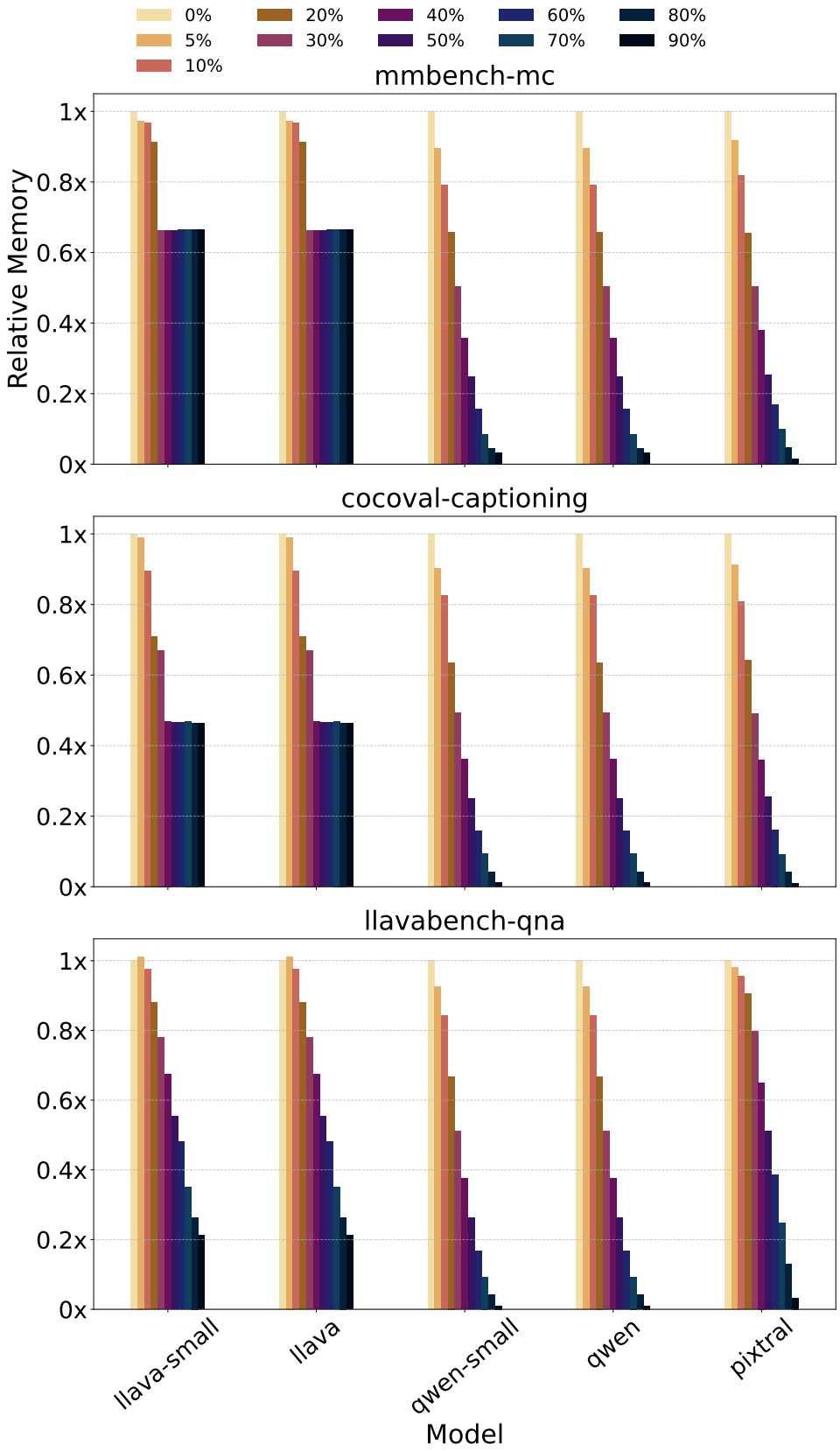


Figure 4.4: *Relative Memory Reduction for All Image Tasks across all Models and all Percentage Reductions*

Some Examples

Let's also provide some qualitative context, demonstrating that the image reduction strategy is a valid system optimization that preserves relative quality, despite the inherent limitations of the model and the ROUGE-L metric.

While the quantitative results confirm that proportional image reduction yields significant latency and memory gains with minimal score degradation, a qualitative analysis of sample outputs is essential. This comparison, using the Pixtral model, highlights the trade-off between optimization and fidelity and exposes the limitations of the ROUGE-L metric in assessing contextual accuracy.

Crucially, the focus of this analysis is not the absolute quality or correctness of the model's responses, but rather how the responses change for the same tasks and samples when the modality input is modified.

(a) (b)

Figure 4.5: *Original Image (a) and 70% Reduced Image (b) from Captioning Dataset*

Commentary on Captioning Example (Table 4.1):

- **Ground-Truth Possible Captions:**
  - A yellow book bus driving down a street.

Table 4.1: *Example 1: COCOVAL Captioning (Descriptive Task) - see Figure 4.5*

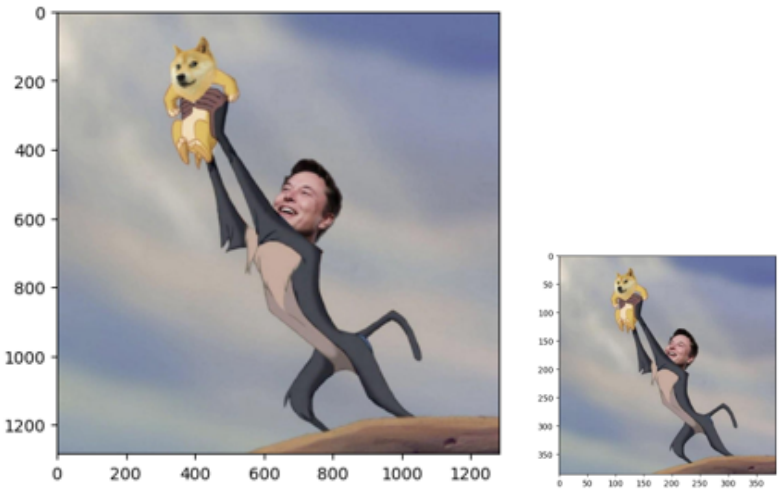
Input	Question	Pixtral Response	ROUGE-L Score
Original Image	Please describe this image in general.	The image shows a yellow and blue bus with the words ””BOOK BUS”” written on the front. The bus has a windshield with windshield wipers, and the front displays the brand ””ISUZU.”” The bus is parked inside a building, and there are lights illuminating the area. The bus appears to be designed for transporting books, possibly for a mobile library or bookstore.	0.295
70% Reduced Image	Please describe this image in general.	The image shows a bus with the words ””Book Bus”” written on the front. The bus is parked in a dimly lit area, possibly a garage or a parking lot. There are several books visible through the front window of the bus. The bus has a yellow and black color scheme.	0.288

  - The big book bus is blue and yellow
  - A blue and yellow ”book bus” is parked on the street
  - The bus is parked and ready to pick up more people
  - There is a blue and yellow bus stopped with the words, Book Bus, on it

- **Content Preservation:** Both captions are highly accurate and descriptive. The reduced image maintains the key semantic entities (”Book Bus,” color scheme, parking location). The 70% reduction causes only a minor drop in the ROUGE-L score (from 0.295 to 0.288).

- **ROUGE-L Limitation:** The high ROUGE-L scores in both cases are good indicators of descriptive quality. However, the metric does not punish the 70% reduced response for incorrectly identifying the color scheme as ”yellow and black”

instead of the original's "yellow and blue"—a minor error potentially due to loss of color fidelity in the reduced image. Despite this, the core message about the bus's purpose is perfectly preserved.



(a) (b)

Figure 4.6: *Original Image (a) and 70% Reduced Image (b) from LLaVABench Dataset*

Commentary on Open-Ended QnA Example (Table 4.2):

- **Ground Truth (GPT-4 Response):** The meme is parodying an iconic scene from The Lion King, where Rafiki holds up Simba to all the other animals while standing on a mountaintop.

- **Model Capability:** It is important to note that the Pixtral model fails to identify the correct iconic scene (The Lion King) even when provided with the original, full-resolution image. The response to the original image incorrectly suggests 'Titanic', while the response to the 70% reduced image incorrectly suggests 'King Kong'.

- **Comparing Modality Impact:** The slight change in the model's incorrect response (from Titanic to King Kong) for the 70% reduced image results in a marginal ROUGE-L change (from 0.216 to 0.202). This minimal change confirms that the misclassification is a matter of the model's inherent reasoning capability and pre-training alignment, not the simple loss of resolution.

Table 4.2: *Example 2: LLaVABench Open-Ended QnA (Descriptive Task) - see Figure 4.6*

Input	Question	Pixtral Response	ROUGE-L Score
Original Image	Which iconic movie scene is being parodied in the meme?	The meme is parodying the iconic scene from the movie 'Titanic' where Jack is holding Rose in the air with one hand while she has her arms and legs spread out, as they both look up at the sky.	0.216
70% Reduced Image	Which iconic movie scene is being parodied in the meme?	The meme is parodying the iconic scene from the movie ""King Kong"" where the giant gorilla, King Kong, is holding a woman in his hand and climbing a tall building. This scene is one of the most recognizable moments in film history, and the meme uses it as a basis for humor by replacing the woman with a cartoon dog, specifically the Doge meme.	0.202

- **Conclusion:** This example underscores that while we observe minor fluctuations in quality, the system optimization of image reduction does not fundamentally alter the reasoning capabilities of the model, confirming its validity as an inference efficiency technique.

4.2 Video Modality Results

The experiments on video workloads focus on analyzing the efficacy of intelligent frame sampling via Scene Change detection—a technique designed to reduce the number of modality tokens ($N_{modality}$) while preserving crucial temporal information. This section quantifies the impact of this strategy on latency, memory footprint, and task quality across various video benchmarks. All results presented for video modality reduction utilize the Scene Change sampling technique, which was identified as optimal in the preceding comparative analysis.

4.2.1 Baseline Latency Analysis

To set the stage for understanding the impact of frame reduction, we first examine the component latency breakdown for all five MLLM architectures when processing video inputs. This initial analysis helps identify the primary bottlenecks within the video processing pipeline.

As illustrated in the baseline video latency plot (Figure 4.7), the latency profiles for video tasks differ significantly from image tasks, primarily due to the sequential nature of video processing and the higher volume of visual information.

- **Dominance of LLM (Decode) Time:** For nearly all models and tasks, the LLM (Decoder) computation time is the overwhelming dominant factor in the total latency. This is particularly evident for the Pixtral-12B model, which exhibits significantly higher LLM latency across all video benchmarks compared to other models. This indicates that processing the sequence of video frames and generating a textual response is highly computationally intensive for the language model backbone.

- **Encoder and Preprocess Contributions:** While substantial, the combined Preprocess and Encoder times (which involve loading video frames, converting them, and transforming them into $N_{modality}$ tokens) represent a smaller proportion of the total latency compared to the LLM phase, especially when the total LLM time is very high. However, they are still significant and contribute noticeably to the overall Time-to-First-Token (TTFT). For example, in videomme-mc for llava-small, the encoder time is a considerable portion.

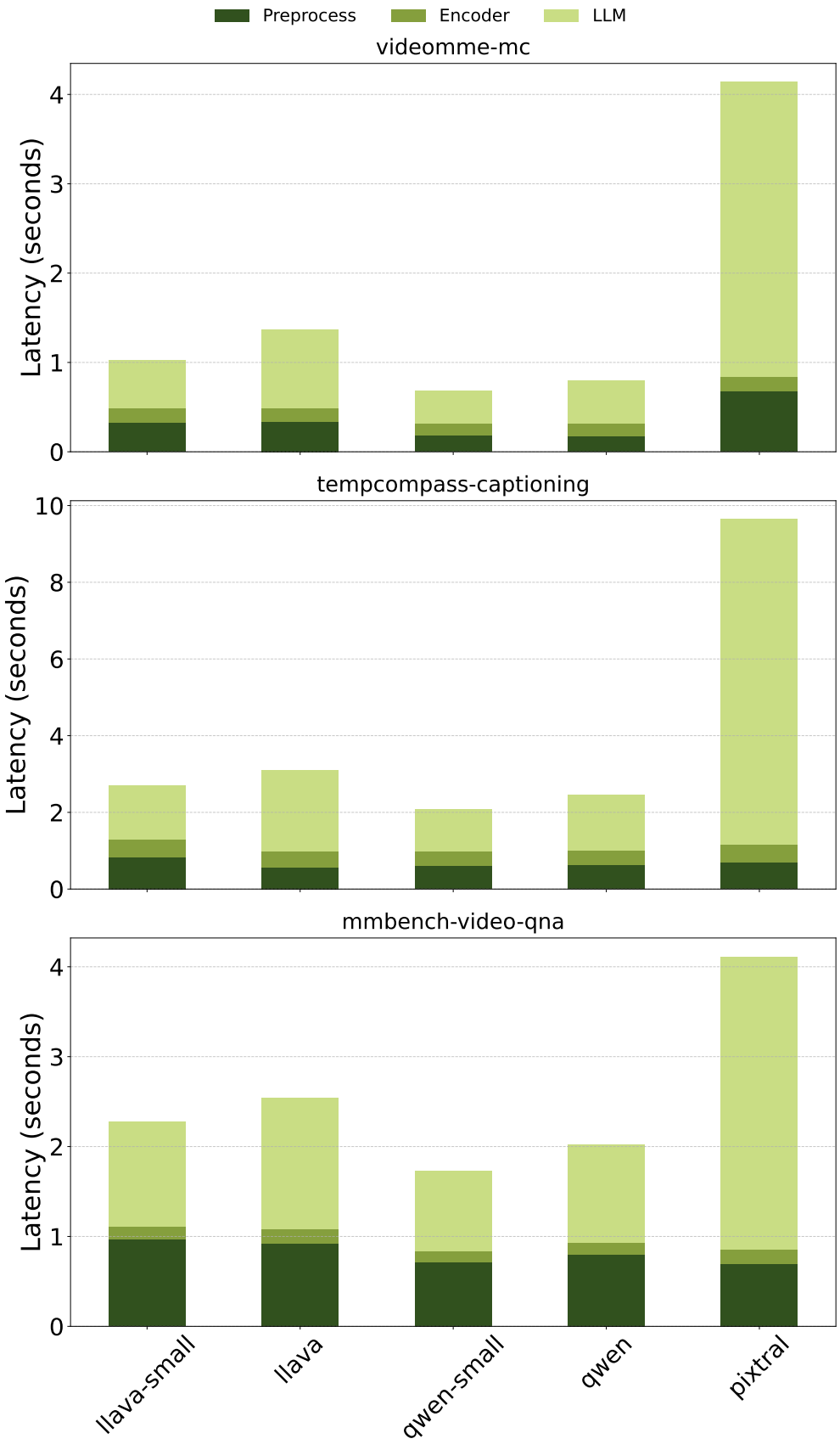


Figure 4.7: *Latency Breakdown for All Video Tasks across all Models*

- **Task Variability:**
  - TempCompass-Captioning generally shows higher total latencies, particularly for pixtral, due to the nature of generating longer, more complex descriptive responses from video.
  - Video-MME-MC and MMBench-Video-QNA also show considerable LLM-dominant latencies, reflecting the cost of complex temporal reasoning.

The analysis confirms that while preprocessing and encoding individual video frames contribute to latency, the language model's processing of the aggregated video tokens and subsequent token generation is the primary bottleneck. Our subsequent analysis will investigate how reducing the number of video frames (and thus N_modality) through Scene Change sampling impacts this LLM-centric latency, along with memory footprint and overall model quality.

4.2.2 Modality Reduction Results

The experiments on video workloads analyzed the impact of intelligent frame sampling via Scene Change detection, a strategy designed to significantly reduce the modality token count ($N_{modality}$) and, consequently, the memory footprint and latency. All results compare performance using the baseline of 64 frames (the maximum and most memory-intensive setting) against reduced frame counts (32, 16, 8, and 4 frames).

The quality metrics (Accuracy for videomme-mc and ROUGE-L for generative tasks) reveal the trade-off between reducing visual information and maintaining model utility (see Figure 4.8).

- **Overall Accuracy:** For the videomme-mc task, the model Accuracy decreases on average by -9.20% across all models and reduced frame numbers. This is a noticeable drop compared to image downsizing but remains acceptable, considering the drastic 93.75% reduction in frames (from 64 to 4). It confirms that while Scene Change is efficient, the model loses some essential temporal detail when the input video signal is severely sparse.
- **Generative Tasks (ROUGE-L):**

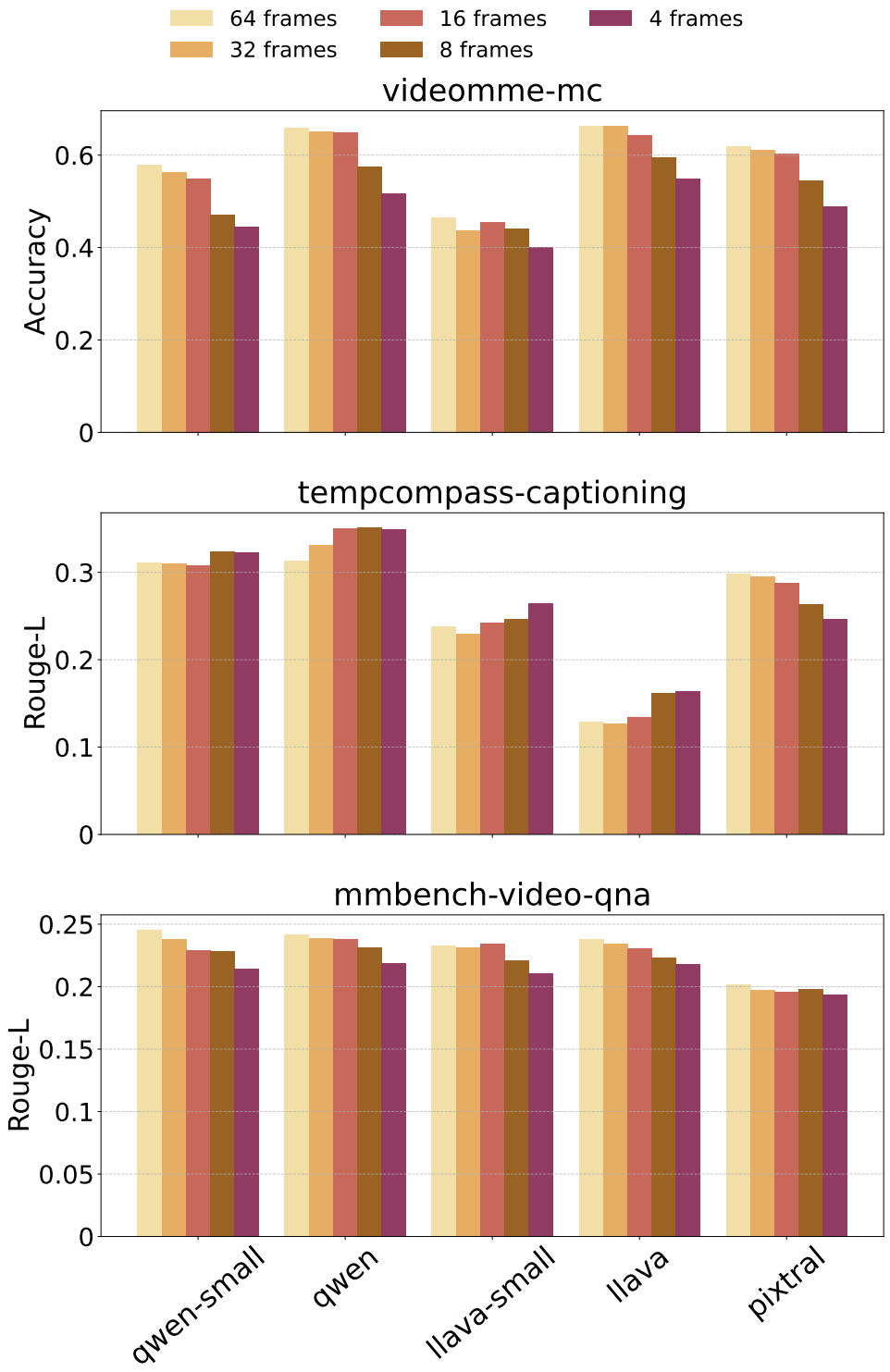


Figure 4.8: *Accuracy and Rouge-L Scores for All Video Tasks across all Models and all Max Frames*

– TempCompass-Captioning: The average ROUGE-L score actually increases by 4.08% across all models and reductions. This positive fluctuation suggests that the Scene Change technique effectively isolates the most semanti-

cally relevant frames, potentially providing a cleaner signal for simple video description than the high redundancy of 64 uniformly sampled frames.

  - MMBench-Video-QNA: This task shows a minimal average ROUGE-L reduction of -4.51%. This suggests that complex video reasoning and open-ended QNA are somewhat more dependent on a denser set of video frames for temporal detail, although the minimal loss validates the Scene Change method's ability to retain context.

- **Conclusion on Quality:** Despite the severe sparsity introduced (down to 4 frames), the quality scores remain robust. The average quality reduction is limited, and the gain in the captioning task highlights the benefit of content-aware sampling over brute-force uniform selection.

The relative latency plot (Figure 4.9) visualizes the reduction in Time-to-First-Token (TTFT), relative to the 64-frame baseline . Since the total input sequence length N_{total} is heavily influenced by the frame count, N_{modality}, this reduction directly confirms the performance gains.

- **Average Latency Gains:** The reduction in latency is dramatic across all tasks and models, far exceeding the gains seen in image downsizing:

  - videomme-mc: Average relative reduction is 0.36x (or a 64% speedup).
  - tempcompass-captioning: Average relative reduction is 0.21x (or a 79% speedup).
  - mmbench-video-qna: Average relative reduction is 0.27x (or a 73% speedup).

- **Significance:** The final latency at the lowest frame count (4 frames) is, on average, less than one-third of the baseline, with the captioning task achieving almost an 80% speedup. This demonstrates the immense cost-benefit of reducing the token count in the prefill phase, which is required to process $N_{modality}$ tokens at $O(N^2)$ complexity.

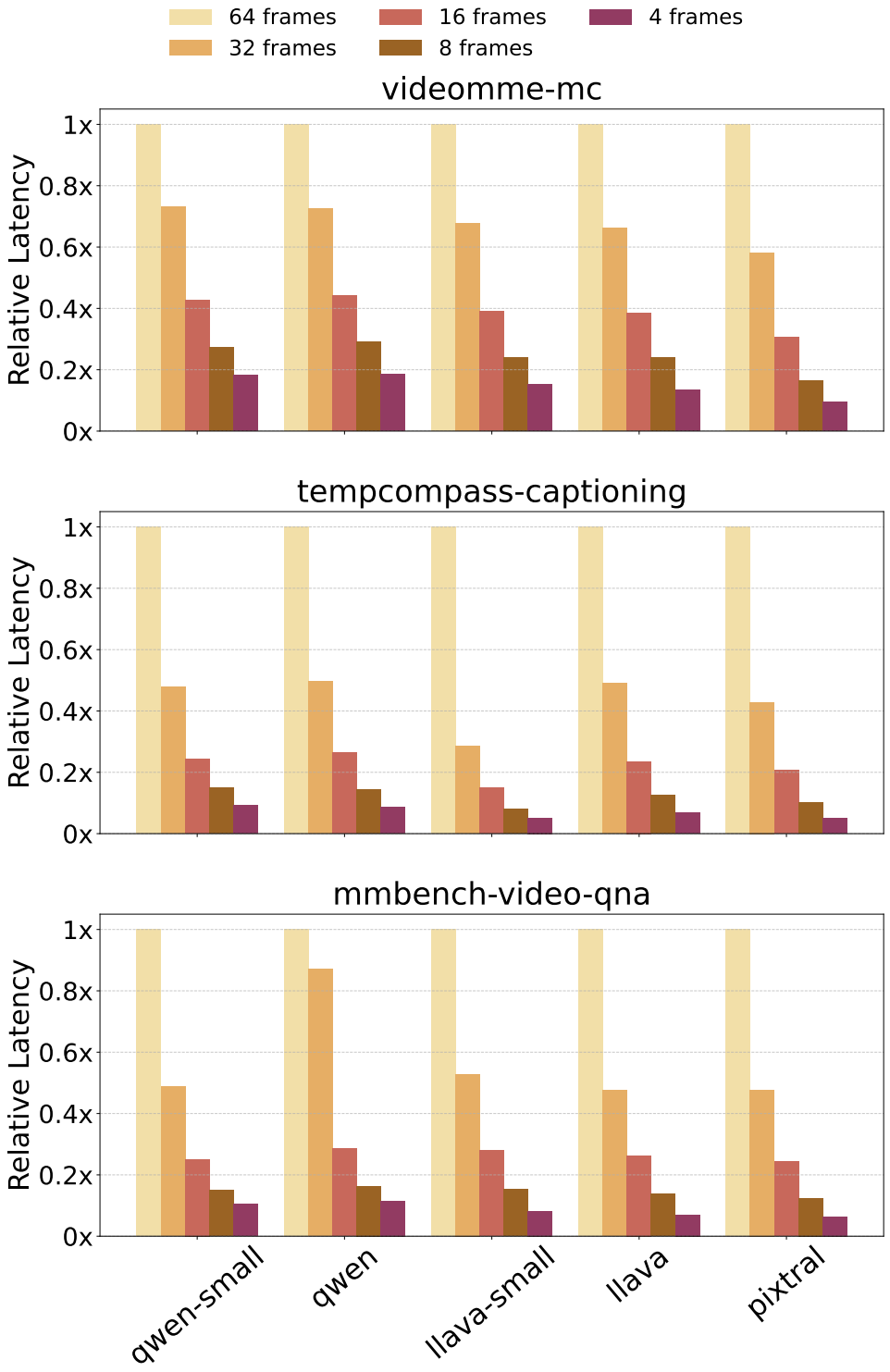


Figure 4.9: *Relative Latency Reduction for All Video Tasks across all Models and all Max Frames*

- **Model Performance:** All models show similar steep decline curves, confirming that the efficiency gain is largely a system-level effect driven by the reduction in

the total number of input tokens being fed into the LLM backbone and processed by vLLM.

The relative memory footprint plot (Figure 4.10) shows the reduction in the total VRAM required to store the KV cache, relative to the 64-frame baseline .

- **Average Memory Reduction:** The memory savings are highly consistent across all tasks, directly correlating with the reduced frame count:
  - videomme-mc: Average relative memory reduction is 0.31x.
  - tempcompass-captioning: Average relative memory reduction is 0.24x.
  - mmbench-video-qna: Average relative memory reduction is 0.26x.
- **Linear Relationship:** The reduction in memory footprint closely follows the reduction in the frame count. Since the KV cache size is directly proportional to the total sequence length (N_{total}), and $N_{modality}$ is the dominant factor in N_{total} for video, cutting the number of frames provides an almost linear reduction in memory overhead.
- **System Impact:** This significant reduction (up to 76% memory saved) drastically reduces the memory pressure on the NVIDIA A100's HBM, allowing vLLM's PagedAttention to serve a much larger number of concurrent requests, thereby maximizing system throughput.

In conclusion, intelligent frame sampling via Scene Change detection provides massive gains in both latency and memory efficiency (up to 79% TTFT reduction and 76% memory reduction) with an acceptable, minimal reduction in model quality, making it a powerful and necessary optimization for serving video-MLLMs.

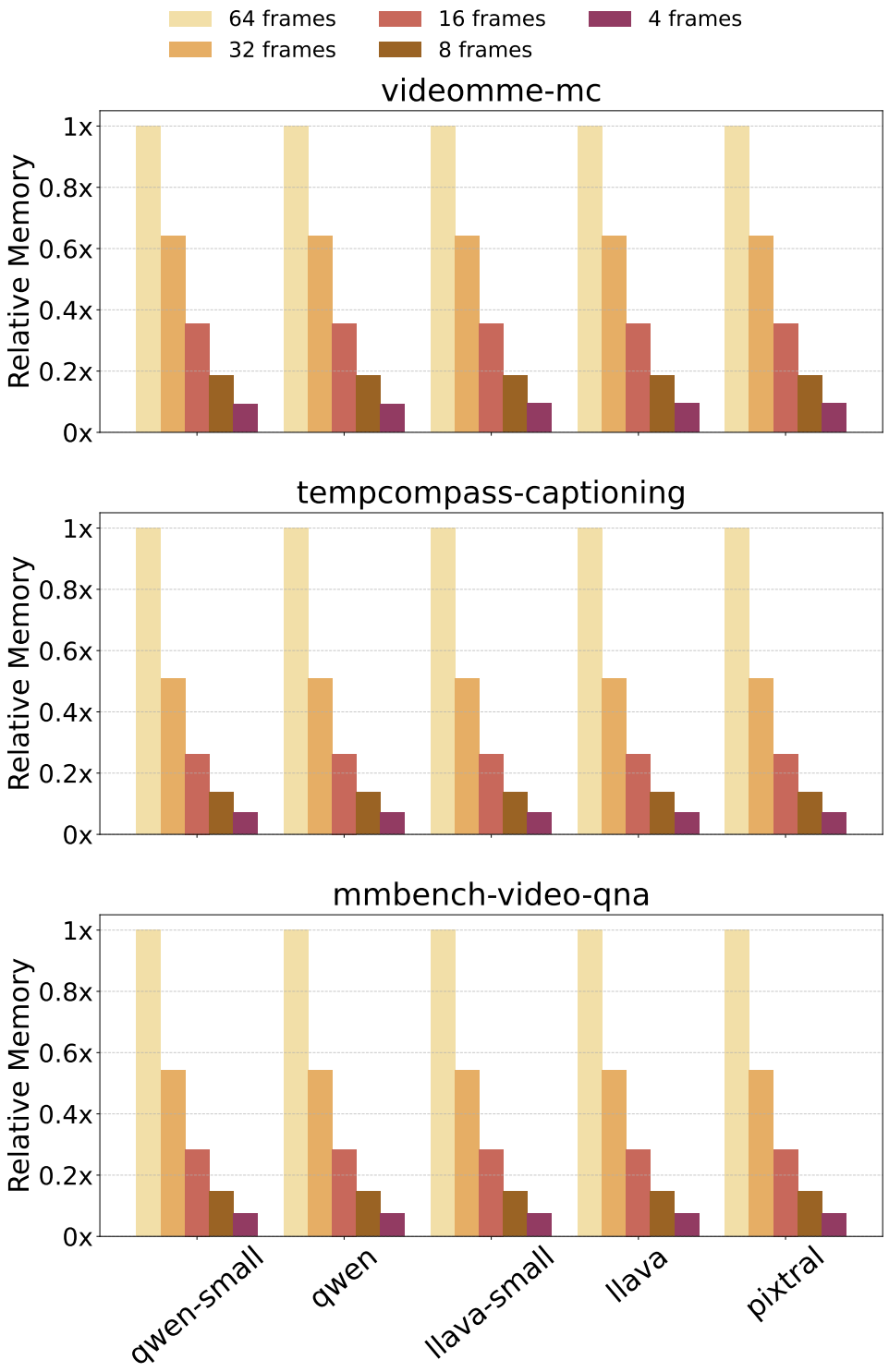


Figure 4.10: *Relative Memory Footprint Reduction for All Video Tasks across all Models and all Max Frames*

5 Conclusion and Future Work

This chapter summarizes the findings and contributions of this thesis regarding the optimization of inference serving for Multimodal Large Language Models (MLLMs), validating modality reduction as a powerful system-level enhancement.

5.1 Summary of Contributions

The goal of this thesis was to empirically validate the effectiveness of system-level modality reduction techniques in mitigating the computational and memory bottlenecks of MLLM inference, specifically the quadratic complexity ($O(N^2)$) of the prefill phase and the resulting Key-Value (KV) cache memory pressure. Using vLLM and a diverse set of MLLMs, we achieved the following core contributions:

- **Validation of Modality Reduction as a System Optimization:** We confirmed that reducing the size and quantity of multimodal inputs (images and video frames) is a highly effective strategy to improve system performance without relying on algorithmic modifications to the LLM decoding process.
- **Quantified Efficiency Gains through Image Downsizing:** Proportional image size reduction demonstrated significant gains in the primary latency metric, Time-to-First-Token (TTFT), and VRAM efficiency. We observed an average TTFT reduction of up to $\mathbf{51}\%$ and an average KV cache memory reduction of up to $\mathbf{53}\%$ on image tasks. Crucially, this was achieved with minimal degradation in quality metrics (average Accuracy decrease of only -3.96%).

- **Optimization via Intelligent Video Frame Sampling:** We validated Scene Change detection as the superior intelligent frame sampling technique. This method achieved massive performance gains by reducing the number of input video frames: an average TTFT reduction of up to **79**% and an average KV cache memory reduction of up to **76**%. This was accomplished while retaining competitive task quality (e.g., a 4.08% ROUGE-L increase in captioning tasks, confirming its ability to select semantically relevant frames).

- **System Capacity Enhancement:** The observed memory savings directly translate to a significantly enhanced system capacity, allowing the NVIDIA A100's 40GB HBM and vLLM's PagedAttention to serve a substantially larger number of concurrent requests, thus maximizing total system throughput.

- **Analysis of MLLM Architecture under Optimization:** The study confirmed that while the Pixtral-12B (MoE) model has the highest baseline memory footprint, its performance scaling under modality reduction followed trends similar to the dense Qwen2-VL and LLaVA-OneVision architectures, validating the reduction technique's effectiveness across distinct MLLM backbones.

5.2 Future Work

The findings of this thesis open several promising avenues for future research in MLLM serving:

- **Dynamic, On-the-Run Optimization:** The ideal outcome of this work is its application in a next-generation serving system. Future research should focus on creating a system that dynamically reduces the size or frame count of modalities on-the-run, using machine learning to predict the optimal quality/performance trade-off for a given request or user profile.

- **Expansion to Other Modalities:** The principle of content-aware reduction could be extended to other sensory inputs. This includes investigating similar sampling techniques for audio datasets (e.g., focusing on speech or event segments) and complex inputs like 3D data.

- **Multi-Modality Integration:** Future work could explore tasks that combine more than two modalities (e.g., text, image, and audio) and study the combinatorial effects of applying distinct reduction techniques simultaneously across different data streams.

- **Hardware and System Generalization:** The analysis was performed on a specific hardware and software stack (NVIDIA A100 and vLLM 0.8.4). Future studies should generalize these findings by testing modality reduction on different GPU architectures (e.g., inference-specific GPUs like the NVIDIA L40) and newer versions of serving frameworks.

Bibliography

[1] Konstantinos Papaioannou and Thaleia Dimitra Doudali. Systems for LLMs Are Old News: Multimodality Is Redefining Everything We Know. EuroSys '25 Poster Session, 2025. Poster. [Online]. Available: `https://2025.eurosys.org/posters/eurosys25posters-paper96.pdf`.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[6] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time, 2017.

[7] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Nee-

lakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: open and efficient foundation language models. arxiv. *arXiv preprint arXiv:2302.13971*, 2, 2023.

[9] AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. Mistral 7b. arxiv 2023. *arXiv preprint arXiv:2310.06825*, 2024.

[10] Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang,

Tianming Liu, and Bao Ge. Understanding llms: A comprehensive overview from training to inference. *Neurocomputing*, 620:129190, 2025.

[11] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. -, 2018.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[14] Philip Gage. A new algorithm for data compression, 1994.

[15] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[16] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from

natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.

[18] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc., 2023.

[19] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikoł aj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736. Curran Associates, Inc., 2022.

[20] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 11 2024.

[21] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[22] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023.

[24] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Icml*, volume 2, page 4, 2021.

[25] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 23–29 Jul 2023.

[26] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Llm inference serving: Survey of recent advances and opportunities. In *2024 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, 2024.

[27] Konstantinos Papaioannou and Thaleia Dimitra Doudali. Improving the efficiency of llm inference serving systems. In Silvina Caino-Lores, Demetris Zeinalipour, Thaleia Dimitra Doudali, David E. Singh, Gracia Ester Martín Garzón, Leonel Sousa, Diego Andrade, Tommaso Cucinotta, Donato D'Ambrosio, Patrick Diehl, Manuel F. Dolz, Admela Jukan, Raffaele Montella, Matteo Nardelli, Marta Garcia-Gasulla, and Sarah Neuwirth, editors, *Euro-Par 2024: Parallel Processing Workshops*, pages 342–347, Cham, 2025. Springer Nature Switzerland.

[28] Konstantinos Papaioannou and Thaleia Dimitra Doudali. The importance of workload choice in evaluating llm inference systems. In *Proceedings of the 4th Workshop on Machine Learning and Systems*, EuroMLSys '24, page 39–46, New York, NY, USA, 2024. Association for Computing Machinery.

[29] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA, 2023. Association for Computing Machinery.

[30] NVIDIA. Tensorrt-llm: A tensorrt toolbox for optimized large language model inference. `https://github.com/NVIDIA/TensorRT-LLM`, 2024. Accessed: 2025-11-09.

[31] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023.

[32] Bin Lin, Chen Zhang, Tao Peng, Hanyu Zhao, Wencong Xiao, Minmin Sun, Anmin Liu, Zhipeng Zhang, Lanbo Li, Xiafei Qiu, Shen Li, Zhigang Ji, Tao Xie, Yong Li, and Wei Lin. Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache, 2024.

[33] Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, Santa Clara, CA, July 2024. USENIX Association.

[34] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31094–31116. PMLR, 23–29 Jul 2023.

[35] Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 139997–140031. Curran Associates, Inc., 2024.

[36] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA, July 2022. USENIX Association.

[37] Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, and Yuxiong He. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference, 2024.

[38] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 117–134, Santa Clara, CA, July 2024. USENIX Association.

[39] Xuanlei Zhao, Bin Jia, Haotian Zhou, Ziming Liu, Shenggan Cheng, and Yang You. Hetegen: Heterogeneous parallel inference for large language models on resource-constrained devices, 2024.

[40] Bingyang Wu, Shengyu Liu, Yinmin Zhong, Peng Sun, Xuanzhe Liu, and Xin Jin. Loongserve: Efficiently serving long-context large language models with elastic sequence parallelism. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, SOSP '24, page 640–654, New York, NY, USA, 2024. Association for Computing Machinery.

[41] Cade Daniel Chen Shen Eric Liang Richard Liaw. How continuous batching enables 23x throughput in llm inference while reducing p50 latency. `https://www.anyscale.com/blog/continuous-batching-llm-inference`, 2023. Accessed: 2025-11-09.

[42] vllm Team. vllm documentation. `https://docs.vllm.ai/en/stable/`, 2024. Accessed: 2025-11-09.

[43] 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, 2024.

[44] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer, 2024.

[45] Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault,

Saurabh Garg, Theophile Gervet, Soham Ghosh, Amélie Héliou, Paul Jacob, Albert Q. Jiang, Kartik Khandelwal, Timothée Lacroix, Guillaume Lample, Diego Las Casas, Thibaut Lavril, Teven Le Scao, Andy Lo, William Marshall, Louis Martin, Arthur Mensch, Pavankumar Muddireddy, Valera Nemychnikova, Marie Pellat, Patrick Von Platen, Nikhil Raghuraman, Baptiste Rozière, Alexandre Sablayrolles, Lucile Saulnier, Romain Sauvestre, Wendy Shang, Roman Soletskyi, Lawrence Stewart, Pierre Stock, Joachim Studnia, Sandeep Subramanian, Sagar Vaze, Thomas Wang, and Sophia Yang. Pixtral 12b, 2024.

[46] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[47] Lior Rokach. *Pattern Classification Using Ensemble Methods*. World Scientific Publishing Co., Inc., USA, 2010.

[48] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player? In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 216–233, Cham, 2025. Springer Nature Switzerland.

[49] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

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

[51] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 24108–24118, June 2025.

[52] Xinyu Fang, Kangrui Mao, Haodong Duan, Xiangyu Zhao, Yining Li, Dahua Lin, and Kai Chen. Mmbench-video: A long-form multi-shot benchmark for holistic video understanding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 89098–89124. Curran Associates, Inc., 2024.

[53] Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. Tempcompass: Do video llms really understand videos?, 2024.

[54] Breakthrough et al. Pyscenedetect: Documentation — contentdetector (scenedetect.detectors.content_detector). `https://www.scenedetect.com/docs/latest/api/detectors.html#module-scenedetect.detectors.content_detector`, 2025. Accessed: 2025-11-10.

[55] Python's Gurus. Farneback algorithm. `https://medium.com/pythons-gurus/farneback-algorithm-50682b8aa2eb`, 2023. Accessed: 2025-11-10.