

Best Practices To Compress Multimodal Large Language Models through Structural Pruning and Recovery

Anonymous authors
Paper under double-blind review

Abstract

Multimodal large language models (MLLMs) are increasingly developed to meet diverse deployment needs, varying in scale and computational demand. While recent research has focused on building MLLMs from Small Language Models (SLMs), these efforts remain limited in flexibility and are still data- and compute-intensive. In this paper, we present the first comprehensive study on flexibly compressing existing MLLMs through structural pruning and recovery training in a data-efficient manner. Hence, we address a critical gap in the literature by empirically analyzing best practices for adapting to specific hardware or resource limitations. Our study investigates pruning and knowledge distillation techniques, examining their impact on downstream performance across various model compression strategies, including pruning paradigms and recovery training schemes. We further investigate the feasibility of performing recovery training using only a small fraction of the available data. Key findings reveal that widthwise pruning is more effective than layerwise pruning in resource-constrained scenarios. For smaller compression ratios, finetuning the multimodal projector alone can restore most performance, while combining finetuning with hidden state knowledge distillation proves most effective across all compression levels. Notably, we demonstrate efficient model downsizing using as little as 5% of the original dataset for moderate compression, which achieves over 95% of the performance compared to using the full dataset. Our paper addresses a critical gap in the literature by empirically analysing the best practices for compressing MLLMs. With our best practices, Bunny-v1.0-3B retains over 95% of its original performance, while LLaVA-v1.5-7B maintains more than 97%, with compression ratios below 30%.

1 Introduction

State-of-the-art multimodal large language models (MLLMs) (Liu et al., 2023; Chu et al., 2023; Chen et al., 2024b) based on Large Language Models (LLMs) (Touvron et al., 2023; Jiang, 2024) require substantial resources. For instance, the LLaVA family (Liu et al., 2023) includes models with parameter counts ranging from 7 to 34 billion. Even those designed to be more memory-efficient, such as Bunny-v1.0-3B (He et al., 2024), still require significant storage, with 3 billion parameters. Reducing the size of these models without compromising performance is crucial for adapting them to diverse deployment scenarios with varying resource constraints.

Despite the growing need for efficient MLLMs, most existing research has focused on building MLLMs on Small Language Models (SLMs) (Zhu et al., 2024a; He et al., 2024; Chu et al., 2023). While these approaches successfully reduce the overall model size, their flexibility is constrained by the fixed size of the underlying SLM. Furthermore, training an SLM from scratch to meet desired specifications is computationally expensive (Chu et al., 2023). Meanwhile, efforts to compress multimodal models have largely focused on task-specific tuning for tasks like visual question answering and image captioning (Wang et al., 2023; Shi et al., 2023). To the best of our knowledge, no previous work has investigated general-purpose model compression for MLLMs.

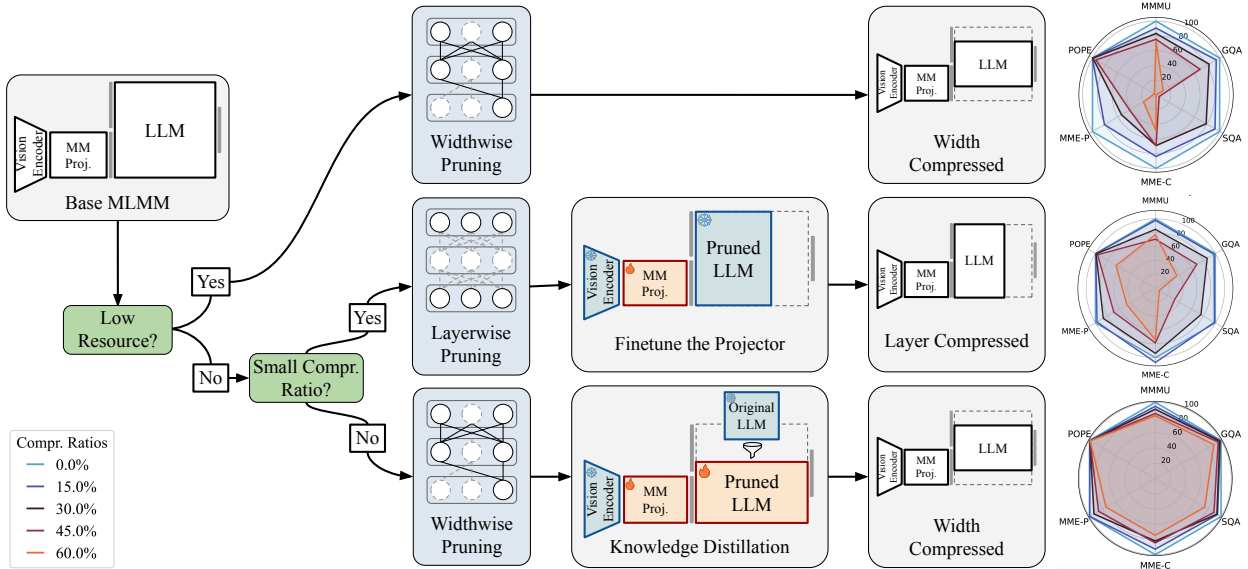


Figure 1: Overview of our best practices for MLLM compression. After evaluating two pruning strategies, i.e., widthwise and layerwise pruning, and multiple recovery strategies, we propose distinct compression approaches for MLLMs. The decision flow highlights the scenarios for applying each method, depending on resource availability and compression ratio requirements. For each approach, we display the resulting compressed model performance across a range of compression ratios (0-60%) on a set of multimodal benchmarks. The spider plots illustrate the retained performance across different tasks, demonstrating each strategy’s effectiveness at various compression levels.

In this work, we aim to uncover the key practices for obtaining effective and compressed MLLMs through structural pruning and recovery training. We adapt and evaluate several techniques designed for compressing LLMs on MLLMs and investigate how different design choices for performance recovery affect the downstream performance of the compressed MLLMs. Our comprehensive empirical study explores several key dimensions: the structural pruning paradigms, the objectives used to restore initial performance, and the amount of data required for effective recovery. Specifically, we examine two MLLMs: a large-scale model (LLaVA-v1.5-7B (Liu et al., 2024)) and a model already optimized for efficiency (Bunny-v1.0-3B (He et al., 2024)). Given that the LLM component dominates the parameter count (95% for LLaVA-v1.5-7B, 86% for Bunny-v1.0-3B), we focus our pruning efforts on it. Specifically, we apply layerwise pruning, which removes entire transformer layers, and widthwise pruning, which reduces attention heads and MLP hidden dimensions. We then evaluate different strategies to recover the potential performance loss: supervised finetuning, knowledge distillation from the original model on the logits distribution, or knowledge distillation on the hidden states. Finally, we investigate how to combine these losses and their effectiveness w.r.t. the amount of available training data.

Our systematic evaluation across different compression ratios led to different key findings:

- **Widthwise pruning is more effective in low-resource scenarios** as it produces an efficient model even without recovery training.
- **With recovery training, layerwise pruning is better for small ratios**, while widthwise pruning usually outperforms it at larger ones (greater than 40%).
- **Finetuning only the multimodal projector is sufficient at small compression ratios**, where pruning has a minimal impact on the language model itself but destroys the multi-modal alignment.
- **The best recovery strategy is supervised finetuning coupled with hidden state based distillation**, consistently achieving the the highest performance across all compression ratios.

- **The higher the pruning ratio, the higher the amount of data needed for recovering the performance.** While with small ratios (less than 50%) even 5% of the data might suffice, this quantity increases for larger ones.

We highlight our key findings in Figure 1. These findings constitute a set of best practices that practitioners can follow when compressing MLLMs and researchers can consider for developing effective pruning strategies without conducting extensive experiments themselves. To ease their exploitation and future studies, we will release our codebase, benchmark as well as the compressed model checkpoints upon acceptance.

2 Related Work

Pruning. Unstructured pruning (Dong et al., 2017; Frankle & Carbin, 2019; Lee et al., 2020; Park et al., 2020; Sanh et al., 2020b; Farina et al., 2024) removes individual weights or neurons. While such approaches can achieve strong compression rates with minimal accuracy trade-offs, they usually require specialized hardware or software for effective acceleration. Structured pruning (Ding et al., 2019; Li et al., 2017; Liu et al., 2021; You et al., 2019) eliminates entire groups of parameters to reduce both the model’s size and its computational overhead. Semi-structured pruning offers a middle ground between structured and unstructured methods by selectively pruning certain model parts. In the context of LLMs, Fang et al. (2023) and Ma et al. (2023) have successfully applied structured pruning, achieving significant sparsity with minimal performance degradation. Xia et al. (2024) targets transformer layers and demonstrates that some layers can be pruned more aggressively without compromising accuracy. Meanwhile, Dery et al. (2024) propose a dynamic pruning strategy that adjusts pruning throughout the training process. In this work, we focus on structure pruning as well as the recovery strategies for MLLMs.

Further Compression Methods. In addition to pruning, techniques like quantization and low-rank factorization are also widely used for model compression. Quantization (Bai et al., 2021; Yao et al., 2022; Zafrir et al., 2019) reduces model size and computational cost by lowering the precision of model parameters, enabling efficient inference with minimal performance loss. Low-rank factorization (Hsu et al., 2022; Hu et al., 2021b; Lan et al., 2020; Ashkboos et al., 2024) compresses models by approximating large weight matrices through the product of smaller matrices, effectively reducing the number of parameters while maintaining most of the model’s capacity. While these methods can offer significant compression, we focus on pruning techniques, which allow for more granular control over the architecture by directly targeting and removing redundant components.

Knowledge distillation (KD) (Hinton et al., 2015) is a standard method for compressing LLMs by transferring knowledge from a large teacher model to a smaller student model (Gou et al., 2021; Sanh et al., 2020a). In NLP classification settings, KD is often applied by having the student model replicate the teacher’s output distribution (Liang et al., 2021; Song et al., 2020; Zhang et al., 2023), hidden states Jiao et al. (2020); Sun et al. (2019), or attention patterns (Wang et al., 2020; 2021), allowing the student to learn from the teacher’s internal representations effectively. For text generation tasks, Xu et al. (2024) provides a comprehensive survey of the role of knowledge distillation in language models. Hsieh et al. (2023) introduce multi-stage distillation, transferring intermediate representations to help the student model capture more detailed features. Gu et al. (2023) propose to replace the forward Kullback-Leibler divergence with a reverse Kullback-Leibler divergence to prevent the student model from overestimating the low-probability regions of the teacher distribution.

Efficient MLLMs. Recent studies (Jin et al., 2024; Zhu et al., 2024b; Lin et al., 2024; Wei et al., 2024) have explored Multimodal Small Language Models (MSLMs). Models such as LLaVA-Phi (Zhu et al., 2024b) utilize pretrained small language models to lower computational costs, while MobileVLM (Chu et al., 2023) concentrates on projector designs to enhance MSLM performance. The Bunny model (He et al., 2024) explores the effects of training data size on performance. Although these approaches reduce model size, they are constrained by the fixed dimensions of the base SLM. Our study specifically addresses methods for customizing the size of existing MLLMs through structured pruning and recovery strategies.

While most multimodal structured compression efforts, such as EfficientVLM (Wang et al., 2023) and UPOP (Shi et al., 2023), focus on task-specific tuning for tasks like visual question answering and image captioning,

general-purpose model compression for MLLMs remains underexplored. Our work addresses this gap by investigating structured pruning techniques and recovery strategies applicable across a variety of multimodal tasks. Unlike previous approaches optimizing models for specific tasks, our study provides general-purpose compression guidelines for MLLMs.

3 Methodology

This section outlines our approach to compressing MLLMs. We first introduce two pruning strategies: layerwise and widthwise pruning. We then describe methods to recover model performance through supervised finetuning and knowledge distillation.

Notation. Given a triplet $\mathbf{X} = \{\mathbf{x}_v, \mathbf{x}_p, \mathbf{x}_r\}$, the objective of an MLLM m_θ , parameterized by $\theta = \{\psi, \phi, \mathbf{W}\}$, is to generate a response \mathbf{x}_r based on an input image \mathbf{x}_v and a text prompt \mathbf{x}_p , such that $m_\theta(\mathbf{x}_v, \mathbf{x}_p) = \mathbf{x}_r$. The MLLM typically consists of a vision encoder $g_\psi(\cdot)$, an LLM $f_\phi(\cdot)$, and a multimodal projector \mathbf{W} aligning the two modalities. The prompt \mathbf{x}_p is tokenized into \mathbf{T}_p , while the vision encoder processes the image \mathbf{x}_v to extract visual features, which are then converted into language embedding tokens \mathbf{T}_v via the multimodal projector:

$$\mathbf{T}_v = \mathbf{W} \cdot g_\psi(\mathbf{x}_v) \quad \text{and} \quad f_\phi(\mathbf{T}_v \odot \mathbf{T}_p) = \mathbf{x}_r. \quad (1)$$

The concatenated visual tokens \mathbf{T}_v and prompt tokens \mathbf{T}_p are fed into the LLM’s M layers, producing hidden states $\{\mathbf{H}_i \in \mathbb{R}^{T \times d}\}_{i=1}^M$, where T is the number of tokens and d is the hidden dimension. Finally, the probabilities $p_{m_\theta}(\mathbf{x}_r | \mathbf{x}_v, \mathbf{x}_p, \tau)$ are computed by passing the final hidden state through the classification head with softmax temperature τ .

3.1 Pruning

Pruning seeks to reduce the number of parameters in a model, thus decreasing its computational cost. In MLLMs, the majority of parameters θ are concentrated in the LLM f_ϕ , so downsizing it can significantly reduce the overall computational burden. The LLM is typically structured along two main dimensions: depth and width. Depth refers to the number of stacked transformer layers, while width pertains to the internal structure of each layer, including the multi-head attention mechanism and the multi-layer perceptron (MLP). In this paper, we explore two pruning strategies to reduce the parameter count in LLMs: *layerwise pruning*, which removes entire transformer layers, and *widthwise pruning*, which eliminates the least important components within each layer.

To determine which layers or components to prune, we draw a small subset of n samples from the the original visual instruct-tuning dataset as the calibration dataset $\mathcal{D} = \{\mathbf{x}_v^j, \mathbf{x}_p^j, \mathbf{x}_r^j\}_{j=1}^n$. The importance of each layer or component is assessed, and those with the lowest importance are pruned.

3.1.1 Layerwise Pruning

Empirical research (Fan et al., 2019; Sajjad et al., 2023) has shown that large transformer models often contain redundant layers, allowing several to be removed with minimal impact on accuracy. To identify and remove these redundant layers, we use the Block Influence (BI) score (Men et al., 2024), which quantifies the importance of layer i through the cosine distance between input \mathbf{H}_i and output hidden states \mathbf{H}_{i+1} . The key assumption is that layers that cause larger changes in hidden states have a greater influence on model performance. The BI score of layer i is then calculated by

$$\text{BI}_i(\mathcal{D}) = 1 - \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, t} \left[\frac{\mathbf{H}_{i,t}^\top \mathbf{H}_{i+1,t}}{\|\mathbf{H}_{i,t}\|_2 \|\mathbf{H}_{i+1,t}\|_2} \right], \quad (2)$$

where $\mathbf{H}_{i,t}$ represents the t^{th} row of \mathbf{H}_i . After calculating the BI scores, the layers are ranked by importance, and those with the lowest scores are pruned.

3.1.2 Widthwise Pruning

Previous research has shown that transformer layers also exhibit width redundancy, meaning that only a subset of attention heads (Voita et al., 2019; Michel et al., 2019) or MLP dimensions (McCarley et al., 2019; Hudson & Manning, 2019) are critical for model performance. To address this, we apply dependency-based structural pruning, which removes redundant widthwise components while minimizing the impact on the model’s performance. Specifically, we first identify groups of interdependent structures and then prune entire groups based on their collective importance.

Following the methods of Fang et al. (2023) and Ma et al. (2023), we begin by constructing dependency relationships within each LLM layer. Let N_i and N_j represent two neurons in the layer, where $\text{In}(N_i)$ and $\text{Out}(N_i)$ represent the neurons connected to N_i as inputs and outputs, respectively. The dependency of neuron N_j on N_i is defined as:

$$N_j \in \text{Out}(N_i) \cap \text{Num}_{\text{In}(N_j)} = 1, \text{ or } N_j \in \text{In}(N_i) \cap \text{Num}_{\text{Out}(N_j)} = 1, \quad (3)$$

where $\text{Num}_{\text{In}(N_i)}$ refers to the number of nodes connected to N_i as inputs and $\text{Num}_{\text{Out}(N_j)}$ is the number of nodes connected to N_j as outputs. If neuron N_i is pruned, all its dependent neurons N_j must also be pruned. This process results in a set of dependency graphs $\mathcal{G} = \{w_i^k\}_{i=1}^M$, where M is the number of structures in the graph and w_i^k represents the k^{th} weight parameter within a structure.

Once the dependency graphs are constructed, we assess their importance at the group level, since all weights within a graph must be pruned together. Group importance is evaluated by comparing the vision-language modeling loss $\mathcal{L}_{CE}(m_\theta(\mathbf{x}_v, \mathbf{x}_q), \mathbf{x}_r)$ on the calibration dataset, both with and without the weight. To efficiently approximate the importance, we apply a Taylor expansion using gradient information. The importance function is given by:

$$I_{w_i^k}(\mathbf{X}) = |\mathcal{L}_{CE}(\mathbf{X}, m_\theta) - \mathcal{L}_{CE}(\mathbf{X}, m_\theta^{w_i^k=0})| \approx \left| \frac{\partial \mathcal{L}_{CE}(\mathbf{X}, m_\theta)}{\partial w_i^k} w_i^k \right|. \quad (4)$$

We then prune the graphs with the lowest group importance I_G :

$$I_G(\mathcal{D}) = \mathbb{E}_{\mathbf{X} \sim \mathcal{D}} \left[\sum_i^M \sum_k I_{w_i^k}(\mathbf{X}) \right]. \quad (5)$$

3.2 Recovery Training

Pruning a large multimodal language model results in performance degradation, affecting both language modeling and cross-modality alignment. To mitigate this, we investigate two recovery training methods: supervised finetuning (Sec. 3.2.1) and knowledge distillation (Sec. 3.2.2). We consider the original teacher model m_θ^T , the pruned student model $m_{\theta'}^S$, and a recovery dataset \mathcal{D} .

3.2.1 Recovery Training with supervised finetuning

A simple yet effective approach to recovery training is supervised finetuning on the original dataset. This method helps counteract performance degradation by allowing the model to adapt its parameters to the modified architecture while taking advantage of the detailed annotations in the original dataset. Here, we first focus on training only the multimodal projector to realign the vision and language spaces. Second, we jointly finetune both the projector and the language model while keeping the vision encoder fixed, as finetuning the vision encoder does not improve performance (Karamcheti et al., 2024). We use the cross-entropy loss for supervised finetuning, denoted as

$$\mathcal{L}_{sft}(m_{\theta'}^S, \mathcal{D}) = \mathbb{E}_{\mathbf{X} \sim \mathcal{D}} [\mathcal{L}_{CE}(m_{\theta'}^S(\mathbf{x}_v, \mathbf{x}_p), \mathbf{x}_r)]. \quad (6)$$

3.2.2 Recovery Training with knowledge distillation

Knowledge distillation (KD) is a method used to transfer knowledge from a large, well-trained model (the teacher) to a smaller or pruned model (the student) (Hinton et al., 2015). This approach allows the pruned

model to regain lost performance by mimicking the decision-making process of the more capable teacher. In our setup, the uncompressed model acts as the teacher, while the pruned model serves as the student. We explore two main strategies, logits-based KD and hidden state based KD, evaluating different loss functions and their trade-offs.

Logits-based KD focuses on aligning the output probability distributions of the pruned model with those of the teacher model. The logits-based KD loss is defined as

$$\mathcal{L}_{logits}(m_{\theta'}^S, m_{\theta}^T, \mathcal{D}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\mathcal{L}_{KD}(p_{m_{\theta}^T}(\mathbf{x}_r | \mathbf{x}_v, \mathbf{x}_p, \tau), p_{m_{\theta'}^S}(\mathbf{x}_r | \mathbf{x}_v, \mathbf{x}_p, \tau)) \right]. \quad (7)$$

We leverage two distinct KD losses to evaluate the differences between these logit distributions. Given the teacher distribution p_{θ} and the student distribution $q_{\theta'}$, the standard KD objective minimizes the approximated forward Kullback–Leibler (KL) divergence between these two distributions, denoted as $\mathcal{L}_{KD}(p_{\theta}, q_{\theta'}) = D_{KL}(p_{\theta} \| q_{\theta'})$. This approach encourages the student distribution to match all the modes of the teacher distribution.

However, minimizing forward KL can lead q_{θ} to assign excessively high probabilities to areas where p has little or no probability mass (Malinin & Gales, 2019). In contrast, Reversed Kullback–Leibler divergence (RKL) minimizes $\mathcal{L}_{KD}(p_{\theta}, q_{\theta'}) = D_{KL}(q_{\theta'} \| p_{\theta})$, encouraging $q_{\theta'}$ to focus on the major modes of p_{θ} while assigning low probabilities to its less significant regions. This helps the student model avoid learning unnecessary long-tail variations of the teacher distribution and instead focus on generating more accurate responses (Gu et al., 2023; Holtzman et al., 2019).

Hidden State Matching involves aligning the pruned model’s intermediate representations (hidden states) $\mathbf{H}_i^{m_{\theta'}^S}$ with the teacher model’s $\mathbf{H}_i^{m_{\theta}^T}$. The corresponding loss for a layer i can be defined as

$$\mathcal{L}_{match}(m_{\theta'}^S, m_{\theta}^T, \mathcal{D}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\mathcal{L}_{feat}(\mathbf{H}_i^{m_{\theta'}^S}, \mathbf{H}_i^{m_{\theta}^T}) \right], \quad (8)$$

where \mathcal{L}_{feat} refers to a feature matching loss. Both Yang et al. (2024) and Popp et al. (2024) suggest that applying a feature-based L2 distillation loss improves the student model’s performance, particularly for pre-trained vision-language models. Consequently, we employ L2 loss as the feature matching loss $\mathcal{L}_{feat} = \|\cdot - \cdot\|_2^2$.

The total loss for recovery training is computed as:

$$\mathcal{L}(m_{\theta'}^S, m_{\theta}^T, \mathcal{D}) = \alpha \mathcal{L}_{sft}(m_{\theta'}^S, \mathcal{D}) + \beta \mathcal{L}_{logits}(m_{\theta'}^S, m_{\theta}^T, \mathcal{D}) + \gamma \mathcal{L}_{match}(m_{\theta'}^S, m_{\theta}^T, \mathcal{D}) \quad (9)$$

where α , β , and γ are the coefficients that balance the contributions of the different loss components.

4 Experiments

In this section, we first introduce our experimental setup and then demonstrate the main findings on model pruning (Sec. 4.1) and performance recovery (Sec. 4.2, Sec. 4.3). We highlight the findings on recovery training using only a small fraction of data (Sec. 4.4) and present the model compression results following our best practices (Sec. 4.5).

Experimental setup. We evaluate pruning and knowledge distillation strategies on both a large-scale MLLM model (LLaVA-v1.5-7B (LLaVA) (Liu et al., 2024)) and a smaller-scale MLLM model (Bunny-v1.0-3B (Bunny) (He et al., 2024)). LLaVA is built upon Vicuna-v1.5 (Chiang et al., 2023) with 6.7 billion parameters, and Bunny is based upon Phi-2 (Jawaheripi et al., 2023) with 2.8 billion parameters. We provide a detailed overview of the model architectures in Appendix A.1.

For both models, we exclusively use their visual instruction tuning datasets: LLaVA-v1-5-mix665k (Liu et al., 2024) for LLaVA and Bunny-695K (He et al., 2024) for Bunny. During pruning, we randomly select 10 samples from the training dataset as the calibration dataset to compute the importance. For recovery training, we experiment with various portions of the original dataset (5%, 10%, 20%, and 100%) for fine-tuning and knowledge distillation. We set the distillation temperature to 2.0 for logits-based distillation and use the final layer representation for hidden state matching (see Appendix A.2).

We evaluate the pruned and recovery-trained models on visual question-answering tasks using GQA (Hudson & Manning, 2019) and SQA-I (Lu et al., 2022), as well as instruction-following tasks with POPE (Li et al., 2023), MME-Cognition, MME-Perception (Yin et al., 2023) and MMMU (Yue et al., 2024). To ensure consistency, we use the lmms-eval suite (Bo et al., 2024) for all evaluations. For clearer comparisons, we calculate the relative performance as a percentage of the original (uncompressed) model’s performance on each benchmark.

4.1 The effect of pruning on the model performance and resources usage

In this section, we begin by exploring the techniques to obtain the best pruned model. We then examine the resulting reductions in memory usage and computational requirements.

Comparison of Pruning Techniques. As illustrated in Figure 2 (blue lines), for both the Bunny and LLaVA model, widthwise pruning consistently outperforms layerwise pruning in terms of model performance after pruning without recovery training. Specifically, for small compression ratios, such as 15%, the Bunny model retains 95% of its performance, while LLaVA retains 93% (see Appendix A.3 for full results). In resource-constrained scenarios, widthwise pruning without recovery training offers an efficient strategy when a small compression ratio is required. However, as the compression ratio increases, both widthwise and layerwise pruned models show significant performance degradation. Overall, widthwise pruning better preserves the model’s structure and information flow, allowing it to keep performance with minimal adjustments, especially at lower compression ratios. The impact of the pruning method on model performance after finetuning both the projector and the LLM is also illustrated in Figure 2 (green lines). For smaller compression ratios (less than 40%), layerwise pruning offers a slight advantage over widthwise pruning, while widthwise pruning delivers better overall performance for larger compression ratios (greater than 40%). This suggests that finetuning plays a crucial role in reconstructing inter-layer connections and reoptimizing layer components.

Best Practice for MMLM Pruning. Widthwise pruning generally proves more effective than layerwise pruning in obtaining the best pruned model. A widthwise pruned model can often be deployed without recovery training when targeting a small compression ratio (less than 20%). Regarding post-finetuning performance, layerwise pruning shows a slight advantage at compression ratios below 30%, whereas widthwise pruning performs marginally better at higher compression ratios.

From compression ratio to resource usage.

Table 1 provides an overview of how different compression ratios impact memory usage and FLOPS for both the Bunny and LLaVA models compressed via widthwise pruning. Memory consumption refers to the allocated GPU memory, while FLOPS are measured using the Calflops codebase¹. The results demonstrate that higher compression ratios consistently lead to both memory and compute reductions. For example, at a 30% compression ratio, we observe a memory reduction of 25% for Bunny and 28% for LLaVA, with a corresponding decrease in FLOPS of 27% for both models. These reductions continue to scale with larger compression ratios; at a 60% compression ratio, memory usage and FLOPS decrease by 50-60%. We observe similar results for layerwise pruning (see Appendix A.4). This indicates that the achieved compressions directly translate into improvements in memory efficiency and computational cost.

Table 1: Memory requirements (Mem.) and FLOPS for the Bunny and LLaVA models at various compression ratios. The models are pruned widthwise. The evaluation is performed in inference mode, where each model is provided with an image and a prompt containing 50 tokens.

Ratio	Bunny		LLaVA	
	Mem. (MiB)	FLOPS (T)	Mem. (MiB)	FLOPS (T)
0%	6,167	4.77	13,546	9.57
15%	5,380	4.14	11,530	8.21
30%	4,597	3.50	9,548	6.89
45%	3,770	2.84	7,470	5.49
60%	2,992	2.20	5,435	4.17

4.2 Supervised Finetuning for Performance Recovery

Compressing the LLM can degrade its language modeling capabilities. More critically, the impact of pruning the LLM decoder on visual understanding and the alignment between vision and language remains largely

¹Calflops codebase: <https://github.com/MrYxJ/calculate-flops.pytorch>

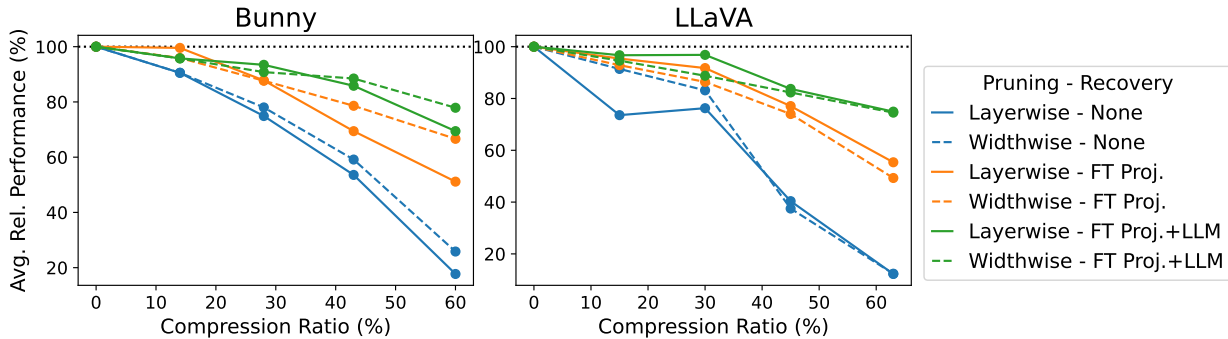


Figure 2: Comparison of pruning and finetuning strategies on two MLLMs, Bunny and LLaVA. The plot shows the average relative performance under three scenarios: pruning only, pruning followed by finetuning the projector, and pruning followed by finetuning both the projector and the LLM. For smaller compression ratios, finetuning only the projector effectively recovers performance. For larger compression ratios, finetuning the projector and the LLM leads to better recovery, indicating the need for broader adjustments as more parameters are pruned.

unexplored. To investigate these effects, we experiment with two approaches: (1) fine-tuning only the multimodal projector and (2) jointly fine-tuning both the projector and the LLM. This allows us to pinpoint the source of performance degradation and assess the extent to which each component contributes to overall model effectiveness.

Following the previous research (Karamcheti et al., 2024), which shows that training the vision encoder may degrade overall model performance, we keep the vision encoder frozen in both setups. To facilitate fast recovery, we employ the low-rank approximation, LoRA (Hu et al., 2021a), while finetuning the LLM.

Finetuning the multimodal projector. As shown in Figure 2 (orange lines), finetuning the multimodal projector significantly restores performance. At lower compression ratios (less than 20%), finetuning only the projector achieves results comparable to jointly finetuning the LLM. For both Bunny and LLaVA, finetuning the projector retains at least 95% of the performance at a compression ratio of 15%. As the compression ratio increases, the loss of language modeling ability becomes more pronounced, making projector-only finetuning insufficient to recover the model’s performance fully. Nevertheless, even at a compression ratio of 60%, only finetuning the multimodal projector can still recover 60 to 80% of the performance by realigning the vision and language inputs. This shows that pruning specific LLM structures in the MLLM can both damage the language modeling ability and introduce modality misalignment, hindering the model’s ability to comprehend visual inputs.

Finetuning both the projector and the LLM. While a significant portion of the recovered performance is due to realigning the visual and textual inputs, we observe consistent gains from additionally finetuning the pruned LLM (green lines in Figure 2), especially at higher compression ratios (greater than 40%). This indicates that the pruned model not only suffers from modality misalignment but also experiences a decline in its language modeling capabilities. We can partly restore these lost capabilities by finetuning the LLM. At a compression ratio of 40%, finetuning both the projector and the LLM restores more than 80% of the original model’s performance. Even at a compression ratio of 60%, finetuning recovers close to 80% of the model’s original performance.

Best Practice for Supervised Finetuning. When a small compression ratio of around 15% is required, finetuning the multimodal projector alone is typically sufficient to recover most of the model’s performance. For higher compression ratios (greater than 40%), incorporating finetuning of the LLM yields additional performance improvements.

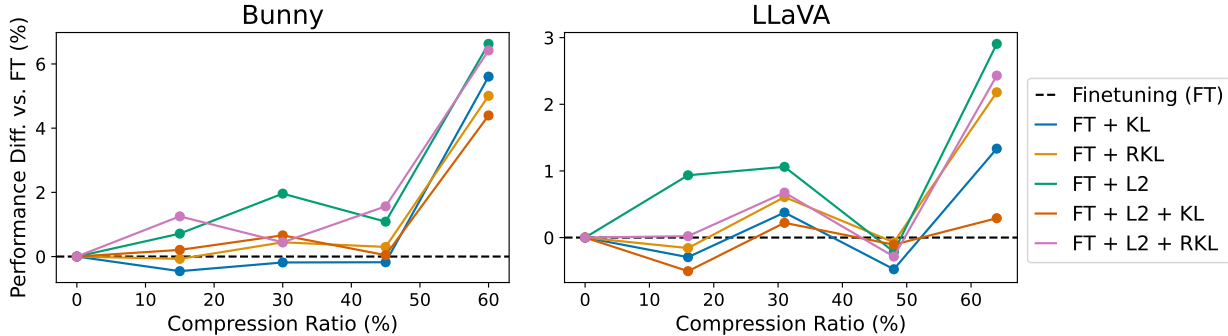


Figure 3: Comparison of different distillation recovery strategies (KL loss, RKL loss, L2 loss, and their combinations) for Bunny and LLaVA models pruned with widthwise pruning. The plot shows the relative performance improvement of each strategy over standard finetuning across various compression ratios. The results demonstrate that distillation helps recover more performance than finetuning alone, with the L2 loss component consistently leading to the largest performance gains.

4.3 Knowledge Distillation for Performance Recovery After Pruning

This section investigates the impact of combining knowledge distillation with finetuning.

As shown in Table 2 for the Bunny model compressed with layerwise pruning, we compare a logit-based approach (RKL) and a hidden state matching strategy (L2), both with and without a finetuning loss component. The results demonstrate that incorporating a supervised finetuning loss significantly enhances and stabilizes distillation performance. For example, when applying only the distillation loss, the L2 and RKL methods can recover 85% of the original performance at a compression ratio of 30%. However, for higher compression ratios, adding the finetuning loss becomes critical in preventing model collapse. At a compression ratio of 60%, combining the finetuning loss with distillation dramatically improves performance—L2 distillation increases from 47.61% to 72.62% and RKL distillation from 12.61% to 69.85%. This pattern is consistently observed across all models and compression techniques evaluated. While knowledge distillation alone can partially recover performance after pruning, its effectiveness is limited without the integration of finetuning.

Table 2: Comparison of distillation strategies with and without finetuning for the Bunny model compressed via layerwise pruning. We show the performance ratio between the compressed model and the original model. Finetuning helps stabilize performance and prevents *model collapse*, especially at higher compression ratios.

Ratio	Bunny				
	FT	L2	L2+FT	RKL	RKL+FT
15%	96.30%	95.51%	99.59%	96.88%	98.70%
30%	94.33%	88.13%	95.03%	92.21%	93.81%
45%	86.70%	56.96%	90.19%	82.57%	88.50%
60%	69.38%	47.61%	72.62%	12.61%	69.85%

Figure 3 compares various distillation strategies based on their relative improvement over finetuning alone when widthwise pruning is applied (see Appendix A.6 for further results on layerwise pruning). Our results indicate that applying the L2 loss to align the hidden states of the student and teacher in the final layer yields the best performance, or at least matches other methods. Unlike logit-based approaches, which require the student to replicate the teacher’s output distribution, the L2 loss method enables the student to better capture the teacher’s feature representations directly, leading to enhanced performance. Additionally, we observe that RKL generally outperforms KL across most compression ratios, a result consistent with the findings of Gu et al. (2024). We perform ablation studies by adjusting the weights of each loss component to evaluate their individual contributions (see 3.2.2). Additional ablation results are in the Appendix A.5.

Best Practice for Knowledge Distillation. Knowledge distillation, particularly when combined with finetuning and using L2 loss to map the intermediate states, delivers the most effective performance recovery after pruning across all compression ratios.

4.4 Data Efficient Recovery

In this section, we investigate the feasibility of performing recovery training using only a small fraction of the available data. Figure 4 shows the models’ performance after recovery training with different portions

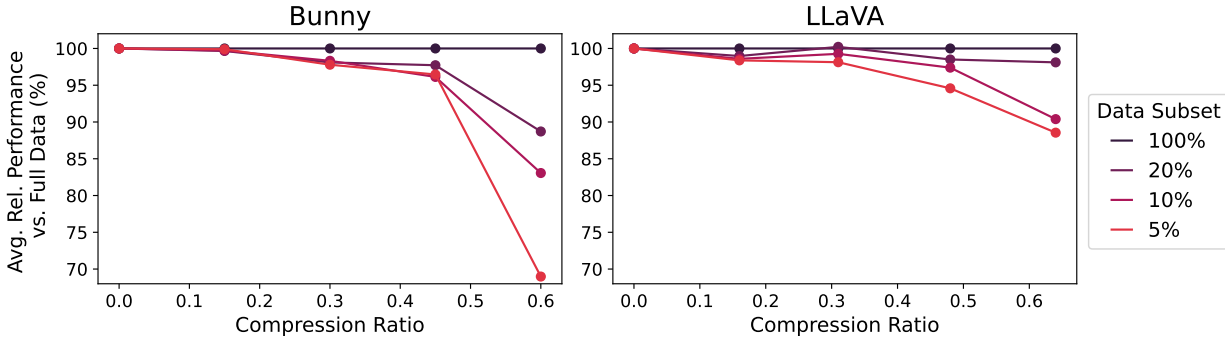


Figure 4: Comparison of recovery performance using different percentages of training data (100%, 20%, 10%, and 5%) for finetuning and distillation after pruning across Bunny and LLaVA models. For smaller compression ratios, even a small percentage of the training data (as low as 5%) is sufficient to recover most of the original performance. However, as the compression ratio increases, more training data is required to achieve higher recovery performance.

of the original dataset relative to training with the full 100%. Both models undergo widthwise pruning and recovery training incorporating RKL and L2 loss functions. Remarkably, for compression ratios below 50%, using just 5% of the original data is sufficient to achieve over 95% of the performance compared to using the full dataset. However, as the compression ratio increases, the amount of data required for effective recovery training also grows. For a compression ratio of 60%, the relative performance drops below 90% for LLaVA and diminishes even further to below 70% for Bunny. Nevertheless, using only a small portion of the training data appears to be a valid option, significantly lowering the required time and cost for compressing and finetuning MLLMs.

Best Practice for Data Efficient Recovery. At small to medium compression ratios less than 50%, using just 5% of the dataset is enough to achieve performance comparable to full data training. However, for compression ratios greater than 50%, full data training becomes necessary to recover performance effectively.

4.5 Model compression results following our best practices

In this section, we summarize our key findings as a set of best practices and highlight model performance achieved by following them. Based on the empirical results from the previous section, we outline the following best practices for compressing MLLMs:

Best Practices for MLLM Compression and Recovery

- **Widthwise pruning is more effective in low-resource settings**, yielding an efficient model even without the need for recovery training.
- **With recovery training**, layerwise pruning excels for smaller compression ratios (below 40%), while widthwise pruning performs better at higher ratios (above 40%).
- **For small compression ratios**, finetuning just the multimodal projector is often sufficient to restore performance, with minimal impact from pruning.
- **For recovery training**, combining finetuning with knowledge distillation of the intermediate representations using L2 loss consistently achieves the highest performance across all compression ratios.
- **Data efficiency** can be significantly boosted, requiring only 5% of the original data to match full-data training results, though full datasets are still needed for high compression ratios.

These guidelines provide insights for researchers aiming to develop new techniques for deploying MLLMs, enabling more effective model customization for specific deployment needs. In Appendix A.7 we offer a detailed comparison of results for both Bunny and LLaVA. These findings underscore the feasibility of

compressing MLLMs without incurring significant performance degradation. We provide qualitative results for the compressed models in Appendix A.8.

5 Discussion

In this section, we discuss the effects of LLM pruning on modality alignment (Sec. 5.1), compare our best practices with quantization and explore their integration (Sec. 5.2), and assess the generalizability of our approach by applying it to a new model (Sec. 5.3). Finally, we discuss the limitations of structural pruning and recovery training as compression techniques, along with potential directions for future work (Sec. 5.4).

5.1 How does pruning LLM impact multi-modal capability?

Since LLMs comprise the majority of parameters in MLLMs, reducing their size can substantially shrink the overall model. However, pruning an LLM presents a dual challenge: it can not only degrade language modeling capabilities but also disrupt the alignment between modalities, impairing the model’s ability to interpret and reason about visual inputs effectively. Our analysis shows that for pruning ratios below 10%, the model retains most of its multimodal functionality. With moderate pruning (up to 15%), modality alignment can still be restored by post-training the multimodal projector. However, beyond this threshold, language modeling degradation becomes more pronounced. At higher compression levels, fine-tuning the projector alone is insufficient to recover performance, making joint training of the LLM necessary to maintain functionality.

5.2 Comparison and combination with quantization

Integrating additional compression techniques, such as quantization, into our framework can further optimize inference time and memory efficiency. In this section, we provide a comparative analysis of structured pruning and quantization, highlighting their complementary strengths when combined. As a representative quantization method, we employ LLM.int8() (Dettmers et al., 2022). As shown in Table 3, LLM.int8() reduces memory usage by 44.5% in the original uncompressed model, while incurring only a minor performance loss of 0.43 percentage points. However, this comes at the cost of a fourfold increase in latency. For LLaVA-6B (layerwise pruned, recovery trained) and LLaVA-5B (widthwise pruned, recovery trained), combining pruning with quantization offers a well-balanced trade-off between memory efficiency and computational latency, mitigating quantization’s overhead while preserving the memory saving. This synergy provides an effective strategy for compressing large models without performance degradation.

Model	Memory(MiB)	Ratio	MMMU	GQA	SQA	MME-C	MME-P	POPE	AVG	Latency
LLaVA-7B	13546		35.10	61.98	68.67	363.21	1511.33	86.99	62.28	105 ms \pm 1.5 ms
LLaVA-7B.int8()	7518		35.20	61.87	68.22	350.71	1508.41	86.54	61.85	398 ms \pm 1.3 ms
LLaVA-6B	11604	15%	35.40	61.17	68.07	328.57	1454.20	86.51	60.82	95 ms \pm 8.1 ms
LLaVA-6B.int8()	6473	15%	35.40	61.17	68.07	328.57	1454.20	86.51	60.82	125 ms \pm 0.9 ms
LLaVA-5B	9548	30%	31.80	60.71	60.54	252.50	1407.08	86.68	56.94	80.7 ms \pm 0.6 ms
LLaVA-5B.int8()	5389	30%	31.60	60.65	60.09	263.57	1410.28	86.78	57.10	141 ms \pm 2.4 ms

Table 3: This table compares different pruning and quantization strategies applied to LLaVA-v1.5-7b, evaluating their effect on memory usage, performance across multiple benchmarks (MMMU, GQA, SQA, MME-C, MME-P, POPE), and inference latency. Quantization using LLM.int8() significantly reduces memory consumption but increases latency, while pruning with recovery training maintains a balance between efficiency and performance. Combining both techniques mitigates quantization overhead while preserving compression benefits.

5.3 Generalizability of the best practices

To assess the generalizability of our best practices, we extended our experiments to Mini-InternVL-Chat-4B-V1-5, which comes from the recent InternVL model family(Chen et al., 2024a)and achieves comparative

results with much larger scale models across benchmarks. It comprises of InternViT-300M-448px as vision encoder, and Phi-3-mini-128k-instruct as the LLM. As shown in Table 4, widthwise pruning outperforms layerwise pruning without recovery training, preserving 97.4% of the original performance at 15% compression compared to 96.7% for layerwise pruning. This reinforces widthwise pruning as the preferred strategy in low-resource scenarios.

We further examined the impact of recovery training, including finetuning the multimodal projector and the LLM, as well as the importance of incorporating knowledge distillation. Table 4 shows that at a 15% compression, projector-only finetuning restores 96.9% of the original performance, while jointly finetuning the projector and LLM improves recovery to 97.8%. At 30% compression, these numbers drop to 75.1% and 86.6%, respectively. Supervised finetuning with hidden state based distillation consistently yields the best results, recovering 98.2% at 15% compression and 87.2% at 30%. These findings confirm our best practices generalize well across architectures, ensuring robustness and broader applicability.

Table 4: Best practices on the model InternVL-Chat-4B. Comparison of different pruning methods, recovery training only multimodal projector (mm) and large language model on Mini-InternVL-Chat-4B-V1-5 after layerwise pruning with different recovery strategies, i.e., supervised finetuning (SFT) and knowledge distillation (KD) on intermediate representations.

Size	Ratio	Pruning	SFT _{mm}	SFT _{all}	KD _{all}	AVG	AVG-%
4B	-	-	-	-	-	72.56	100%
3.5B	15%	Layerwise	-	-	-	70.15	96.68%
3.5B	15%	Widthwise	-	-	-	70.70	97.44%
3.5B	15%	Layerwise	✓	✓	-	70.15	96.68%
3.5B	15%	Layerwise	-	✓	-	70.96	97.80%
3.5B	15%	Layerwise	-	✓	✓	71.23	98.16%
3B	30%	Layerwise	✓	✓	-	43.94	60.56%
3B	30%	Layerwise	-	✓	-	62.86	86.64%
3B	30%	Layerwise	-	✓	✓	63.29	87.23%

5.4 Limitation and future work

Our experiments demonstrate that structural pruning with recovery training enables effective model compression with minimal performance degradation at moderate compression ratios (up to 30%). However, beyond this threshold, performance loss becomes increasingly difficult to recover, indicating fundamental limitations in the model’s capacity to retain essential information. This suggests that for applications requiring more aggressive compression, training MLLMs from smaller pre-trained SLMs may be a more viable alternative than extreme pruning of large models.

This study establishes best practices based on experiments with two representative models (LLaVA-v1.5-7B and Bunny-v1.0-3B) and confirms their generalizability with InternVL. While these models offer valuable insights, extending our analysis to a broader range of architectures and scales would further strengthen our findings. Future work will explore larger and smaller MLLMs and alternative recovery strategies to assess whether our compression strategies remain effective across diverse settings.

6 Conclusion

In this work, we investigated efficient compression techniques for MLLMs, focusing on two key pruning strategies: width and layerwise. Our study assessed the impact of these strategies on model performance, both before and after recovery training, across various compression ratios. We further explored recovery methods such as supervised finetuning and knowledge distillation to address performance degradation caused by pruning. We formulate our findings as best practices, which offer practical guidelines for optimizing MLLMs, enabling a balance between model size, performance, and data efficiency to meet specific deployment resource constraints. Due to computational constraints, in this work we focused on two representative pruning techniques on three different models. Future work could extend these best practices to include a broader range of pruning techniques and models to further refine these strategies.

References

- Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoeffler, and James Hensman. Slicept: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.
- Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization, 2021. URL <https://arxiv.org/abs/2012.15701>.
- Li* Bo, Zhang* Peiyuan, Zhang* Kaichen, Pu* Fanyi, Du Xinrun, Dong Yuhao, Liu Haotian, Zhang Yuanhan, Zhang Ge, Li Chunyuan, and Liu Ziwei. Lmms-eval: Accelerating the development of large multimodal models, March 2024. URL <https://github.com/EvolvingLMMS-Lab/lmms-eval>.
- Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024a.
- Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24185–24198, 2024b.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*, 2023.
- Lucio Dery, Steven Kolawole, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody prune now: Structured pruning of llms with only forward passes, 2024. URL <https://arxiv.org/abs/2402.05406>.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.
- Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure, 2019. URL <https://arxiv.org/abs/1904.03837>.
- Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon, 2017. URL <https://arxiv.org/abs/1705.07565>.
- Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.
- Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16091–16101, 2023.
- Matteo Farina, Massimiliano Mancini, Elia Cunegatti, Gaowen Liu, Giovanni Iacca, and Elisa Ricci. Multi-flow: Shifting towards task-agnostic vision-language pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16185–16195, 2024.
- Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL <https://arxiv.org/abs/1803.03635>.
- Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, March 2021. ISSN 1573-1405. doi: 10.1007/s11263-021-01453-z. URL <http://dx.doi.org/10.1007/s11263-021-01453-z>.

- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models, 2024. URL <https://arxiv.org/abs/2306.08543>.
- Muyang He, Yexin Liu, Boya Wu, Jianhao Yuan, Yueze Wang, Tiejun Huang, and Bo Zhao. Efficient multimodal learning from data-centric perspective. *arXiv preprint arXiv:2402.11530*, 2024.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL <https://arxiv.org/abs/1503.02531>.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.
- Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023. URL <https://arxiv.org/abs/2305.02301>.
- Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization, 2022. URL <https://arxiv.org/abs/2207.00112>.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021a.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021b. URL <https://arxiv.org/abs/2106.09685>.
- Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.
- Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 2023.
- Fengqing Jiang. Identifying and mitigating vulnerabilities in llm-integrated applications. Master’s thesis, University of Washington, 2024.
- Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding, 2020. URL <https://arxiv.org/abs/1909.10351>.
- Yizhang Jin, Jian Li, Yexin Liu, Tianjun Gu, Kai Wu, Zhengkai Jiang, Muyang He, Bo Zhao, Xin Tan, Zhenye Gan, et al. Efficient multimodal large language models: A survey. *arXiv preprint arXiv:2405.10739*, 2024.
- Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020. URL <https://arxiv.org/abs/1909.11942>.
- Namhoon Lee, Thalaisyasingam Ajanthan, Stephen Gould, and Philip H. S. Torr. A signal propagation perspective for pruning neural networks at initialization, 2020. URL <https://arxiv.org/abs/1906.06307>.

- Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017. URL <https://arxiv.org/abs/1608.08710>.
- Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.
- Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. Mixkd: Towards efficient distillation of large-scale language models, 2021. URL <https://arxiv.org/abs/2011.00593>.
- Bin Lin, Zhenyu Tang, Yang Ye, Jiayi Cui, Bin Zhu, Peng Jin, Junwu Zhang, Munan Ning, and Li Yuan. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*, 2024.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL <https://arxiv.org/abs/2304.08485>.
- Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024.
- Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression, 2021. URL <https://arxiv.org/abs/2108.00708>.
- Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.
- Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *Advances in neural information processing systems*, 32, 2019.
- JS McCarley, Rishav Chakravarti, and Avirup Sil. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360*, 2019.
- Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.
- Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.
- Sejun Park, Jaeho Lee, Sangwoo Mo, and Jinwoo Shin. Lookahead: A far-sighted alternative of magnitude-based pruning, 2020. URL <https://arxiv.org/abs/2002.04809>.
- Niclas Popp, Jan Hendrik Metzen, and Matthias Hein. Zero-shot distillation for image encoders: How to make effective use of synthetic data. *arXiv preprint arXiv:2404.16637*, 2024.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.
- Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020a. URL <https://arxiv.org/abs/1910.01108>.

- Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning, 2020b. URL <https://arxiv.org/abs/2005.07683>.
- Dachuan Shi, Chaofan Tao, Ying Jin, Zhendong Yang, Chun Yuan, and Jiaqi Wang. Upop: Unified and progressive pruning for compressing vision-language transformers. In *International Conference on Machine Learning*, pp. 31292–31311. PMLR, 2023.
- Kaitao Song, Hao Sun, Xu Tan, Tao Qin, Jianfeng Lu, Hongzhi Liu, and Tie-Yan Liu. Lightpaff: A two-stage distillation framework for pre-training and fine-tuning, 2020. URL <https://arxiv.org/abs/2004.12817>.
- Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression, 2019. URL <https://arxiv.org/abs/1908.09355>.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.
- Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13899–13913, 2023.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL <https://arxiv.org/abs/2002.10957>.
- Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers, 2021. URL <https://arxiv.org/abs/2012.15828>.
- Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, En Yu, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Small language model meets with reinforced vision vocabulary. *arXiv preprint arXiv:2401.12503*, 2024.
- Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning, 2024. URL <https://arxiv.org/abs/2310.06694>.
- Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL <https://arxiv.org/abs/2402.13116>.
- Chuangang Yang, Zhulin An, Libo Huang, Junyu Bi, Xinqiang Yu, Han Yang, Boyu Diao, and Yongjun Xu. Clip-kd: An empirical study of clip model distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15952–15962, 2024.
- Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers, 2022. URL <https://arxiv.org/abs/2206.01861>.
- Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023.
- Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks, 2019. URL <https://arxiv.org/abs/1909.08174>.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhua Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*, 2024.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*. IEEE, December 2019. doi: 10.1109/emc2-nips53020.2019.00016. URL <http://dx.doi.org/10.1109/EMC2-NIPS53020.2019.00016>.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Rongzhi Zhang, Jiaming Shen, Tianqi Liu, Jialu Liu, Michael Bendersky, Marc Najork, and Chao Zhang. Do not blindly imitate the teacher: Using perturbed loss for knowledge distillation, 2023. URL <https://arxiv.org/abs/2305.05010>.

Minjie Zhu, Yichen Zhu, Xin Liu, Ning Liu, Zhiyuan Xu, Chaomin Shen, Yaxin Peng, Zhicai Ou, Feifei Feng, and Jian Tang. A comprehensive overhaul of multimodal assistant with small language models. *arXiv preprint arXiv:2403.06199*, 2024a.

Yichen Zhu, Minjie Zhu, Ning Liu, Zhiyuan Xu, and Yaxin Peng. Llava-phi: Efficient multi-modal assistant with small language model. In *Proceedings of the 1st International Workshop on Efficient Multimedia Computing under Limited*, pp. 18–22, 2024b.

A Appendix

A.1 Model architecture of the LLaVA and Bunny models used in the main experiments

Table 5 outlines the architectures of the Bunny and LLaVA models. LLaVA-v1.5-7B employs CLIP-ViT-L (Radford et al., 2021) as the vision encoder and Vicuna-v1.5 Chiang et al., 2023 as the language decoder, while Bunny-v1.0-3B utilizes SigLIP-SO (Zhai et al., 2023) as the vision encoder and Phi-2 (Jawaheripi et al., 2023) as the language decoder. Both models leverage MLP layers to align the vision and language modalities.

Model	Parameters	Vision Encoder	Multimodal Projector	Language Decoder
LLaVA-v1.5-7B	7.0B	CLIP-ViT-L (0.3B)	mlp2x-gelu (0.01B)	Vicuna-v1.5 (6.7B)
Bunny-v1.0-3B	3.2B	SigLIP-SO (0.4B)	mlp2x-gelu (0.02B)	Phi-2 (2.8B)

Table 5: Architecture details of the uncompressed models. We present the number of parameters, along with the vision encoder, multimodal projector and the language decoder of the models included in our study.

A.2 Implementation details of hidden state matching

To determine which LLM layers’ hidden states to map between the pruned and unpruned models, we explore three options: matching the last layer, the last two, and the last three layers. Table 6 shows that matching only the last layer’s hidden state yields the best performance.

A.3 Detailed results for two pruning techniques

Table 7 presents the model performance for widthwise and layerwise pruning. For both of the Bunny and LLaVA models, widthwise pruning consistently outperforms layerwise pruning. This performance gap widens as the compression ratio increases, with widthwise pruning showing a more significant advantage at higher compression ratios.

Ratio	Layer-1	Layer-1,2	Layer-1,2,3
12.8%	95.34%	95.17%	96.25%
25.5%	91.02%	90.48%	90.97%
39.0%	87.08%	86.12%	84.84%
51.8%	75.25%	72.56%	72.87%

Table 6: Results for recovering widthwise pruned Bunny with hidden state mapping. We compare the relative performance for mapping the last layer (layer-1), the last two layers (layer-1,2), and the last three layers (layer-1,2,3). By only mapping the last LLM layer the best performance is achieved.

Method	Size	PruneRatio	MMMU	GQA	SQA	MME-C	MME-P	POPE	AVG	AVG-%
LLaVA-v1.5-7B	7.0B		35.10	61.98	68.67	363.21	1511.33	86.99	62.28	100.00%
Width-wise	6.3B	15%	32.40	59.34	63.21	268.93	1432.47	86.57	57.79	92.79%
	5.5B	30%	31.00	52.59	54.29	253.21	1174.93	86.29	52.43	84.17%
	4.8B	45%	27.60	20.86	12.10	70.00	347.45	45.96	22.11	35.49%
	4.0B	60%	23.30	0.43	0.40	2.14	19.24	3.94	4.88	7.84%
Depth-wise	6.3B	15%	31.80	42.77	55.23	202.14	701.83	86.38	46.09	74.00%
	5.5B	30%	32.70	42.18	59.64	210.71	921.88	78.69	47.61	76.43%
	4.8B	45%	26.90	14.39	3.82	132.86	616.63	51.69	24.04	38.60%
	4.0B	60%	25.80	0.00	0.00	0.00	0.00	0.00	4.30	6.90%
Bunny-v10-3B	3.2B		34.10	54.72	70.70	289.30	1487.71	87.82	59.65	100.00%
Width-wise	2.8B	15%	30.90	51.83	65.64	242.50	1207.85	87.94	54.50	95.48%
	2.5B	30%	28.40	45.65	55.73	199.64	807.95	87.13	47.04	87.57%
	2.0B	45%	25.70	37.92	3.42	200.00	618.25	83.12	34.35	60.66%
	1.6B	60%	24.80	6.12	0.00	141.07	293.23	2.34	10.93	13.52%
Depth-wise	2.8B	15%	33.80	29.42	69.66	271.43	1456.41	87.91	54.59	91.52%
	2.5B	30%	29.00	24.77	28.76	272.86	1273.34	86.50	44.47	74.55%
	2.0B	45%	23.90	16.85	3.47	191.43	867.37	80.09	31.94	53.54%
	1.6B	60%	26.60	0.02	17.15	0.71	55.92	0.02	7.78	13.04%

Table 7: Pruning results for LLaVA-v1.5-7B and Bunny-v1-3B. Size is the number of total parameters of the model, while the compression ratio (Ratio) indicates the proportion of remaining language model parameters compared to the pre-pruning state. For both models, width-wise pruning results in better performance without finetuning compared to depth-wise pruning.

A.4 Detailed results on the efficiency of the pruned models

For the models pruned by layerwise method, we also assess their memory consumption as FLOPs. Memory consumption refers to the allocated GPU memory, while FLOPs are measured using the Calflops codebase². The results in Table 8 show the same trend as widthwise pruning, indicating that the achieved compressions directly translate into improvements in memory efficiency and computational cost for both widthwise and layerwise pruning.

A.5 Is SFT needed for KD?

Figure 5 and Figure 6 compare logits-based knowledge distillation (represented by RKL) and hidden

Ratio	Bunny		LLaVA	
	Mem. (MiB)	FLOPS (T)	Mem. (MiB)	FLOPS (T)
0%	6,167	4.77	13,546	9.57
15%	5,411	4.16	11,604	8.03
30%	4,659	3.56	9,664	6.92
45%	3,907	2.95	7,724	5.55
60%	3,006	2.22	5,496	3.9

Table 8: Memory requirements (Mem.) and FLOPS for the Bunny and LLaVA models at various compression ratios. The models are pruned layerwise. The evaluation is performed in inference mode, where each model is provided with an image and a prompt containing 50 tokens.

²Calflops codebase: <https://github.com/MrYxJ/calculate-flops.pytorch>

state matching-based knowledge distillation (represented by L2 loss), with and without supervised fine-tuning, following widthwise and layerwise pruning, respectively. While knowledge distillation alone helps in recovering performance post-pruning, it remains less effective than supervised fine-tuning. However, when combined with supervised fine-tuning, it results in superior performance.

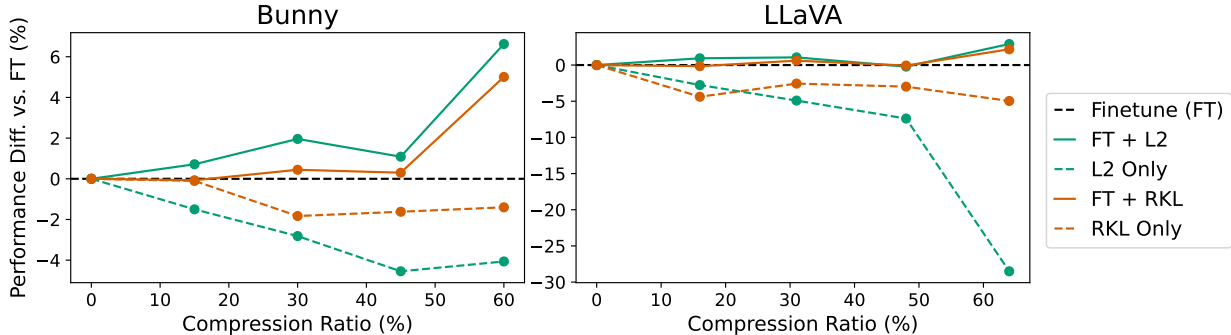


Figure 5: Comparison of L2 and RKL distillation strategies with and without additional fine-tuning loss for Bunny and Llava models compressed by widthwise pruning. The plot shows performance differences relative to standard fine-tuning across varying compression ratios.

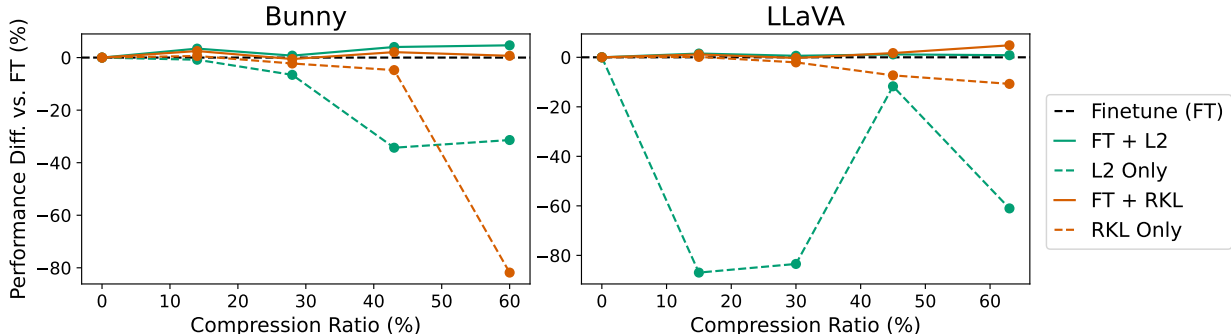


Figure 6: Comparison of L2 and RKL distillation strategies with and without additional fine-tuning loss for Bunny and Llava models compressed by layerwise pruning. The plot shows performance differences relative to standard fine-tuning across varying compression ratios.

A.6 Which KD strategy to use after layerwise pruning?

Figure 7 compares various distillation strategies based on their relative improvement over finetuning alone after layerwise pruning. Similar to the results after widthwise pruning, applying hidden states matching yields the best performance, or at least matches other methods. The trend that RKL generally outperforms KL is also observed here.

A.7 Detailed model compression results following our best practices

This section provides detailed numerical results of the model performance following our best practices. To illustrate the performance at different compression ratios, Table 9 offers a detailed comparison of results for both Bunny and LLaVA across various multimodal benchmarks. The results show that, with compression ratios below 30%, Bunny retains over 95% of its original performance, while LLaVA maintains more than 97%. Even at higher compression ratios, up to 60%, our best practices preserve an average performance

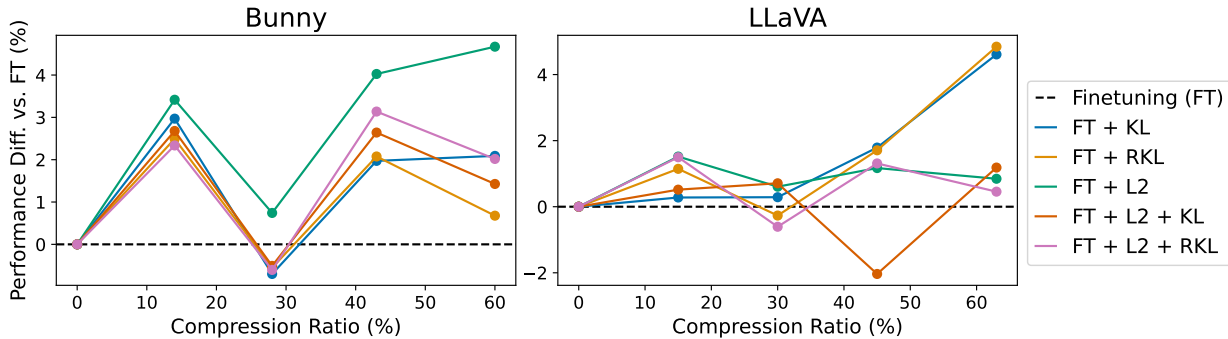


Figure 7: Comparison of different distillation recovery strategies (KL loss, RKL loss, L2 loss, and their combinations) for Bunny and LLaVA models pruned with layerwise pruning. The plot shows the relative performance improvement of each strategy over standard fine-tuning across various compression ratios. The results demonstrate that distillation helps recover more performance than fine-tuning alone, with the L2 loss component consistently leading to the largest performance gains, particularly at higher compression ratios.

of 83% for Bunny and 78% for LLaVA. These findings underscore the feasibility of compressing MLLMs without incurring significant performance degradation.

A.8 Qualitative results of the compressed models

In this section, we present some qualitative results of the compressed models. Table 10 presents the qualitative evaluation results of the compressed LLaVA models. Despite undergoing compression, these models exhibit a remarkable capacity for understanding and processing visual inputs with high accuracy. They effectively analyze images and generate rich, detailed textual descriptions. This demonstrates that compression does not significantly compromise their ability to comprehend complex visual information. Instead, the models maintain strong performance, producing coherent and contextually relevant outputs.

Table 9: Performance of the best compressed models. The size is the number of total parameters of the model, while the ratio, short for compression ratio, indicates the proportion of remaining LLM parameters compared to the pre-pruning state. When the compression ratio (Ratio) is below 40%, we apply depthwise pruning. For ratios above 40%, we use widthwise pruning. During the recovery phase, we employ supervised finetuning combined with L2 loss to match the hidden states. For both Bunny and LLaVA, 95% performance is retained if the compression ratio is smaller than 40%.

Method	Size	Ratio	MMMU	GQA	SQA	MME-C	MME-P	POPE	AVG	AVG-%
Bunny-v1.0-3B										
	3.2B	0%	34.10	54.72	70.70	289.30	1487.71	87.82	59.65	100.00%
Depth+FT+L2	2.8B	15%	33.00	54.56	70.00	304.29	1457.06	87.97	59.40	99.59%
Depth+FT+L2	2.5B	30%	32.30	53.08	68.12	252.50	1349.91	87.53	56.68	95.03%
Width+FT+L2	2.0B	45%	29.10	52.31	63.06	244.64	1281.66	87.09	54.37	91.15%
Width+FT+L2	1.6B	60%	28.10	48.72	53.20	216.07	1115.33	86.73	49.92	83.69%
LLaVA-v1.5-7B										
	7.0B	0%	35.10	61.98	68.67	363.21	1511.33	86.99	62.28	100.00%
Depth+FT+L2	6.3B	15%	36.40	61.20	68.42	337.86	1442.35	86.94	61.22	98.29%
Depth+FT+L2	5.5B	30%	36.00	60.34	68.82	318.57	1496.60	85.98	60.96	97.88%
Width+FT+L2	3.8B	45%	30.80	57.74	52.90	215.00	1191.17	85.74	52.27	83.92%
Width+FT+L2	2.8B	60%	27.70	52.32	46.26	211.79	1085.97	84.06	48.52	77.90%


Model	Response
	<p>Prompt: A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user’s questions. USER: <i>< image ></i> What is funny in the image?</p>
LLaVA-6.3B	<p>The image is funny because it features a dog wearing a Renaissance-style dress and a bandana, posing as if it’s posing for a painting. This is humorous because it is a playful and unconventional representation of a dog, which is not something you would typically see in a painting or a Renaissance-style dress. The dog’s pose and attire create a sense of humor and absurdity, making the image entertaining and amusing to viewers.</p>
LLaVA-5.5B	<p>The image is funny because it features a dog dressed up in a human-like attire, wearing a long black dress and a hat. This unusual and amusing scene is not something you would typically see in real life, as dogs don’t wear clothes or accessories like humans. The dog’s expression and the way it is posing for the picture further contribute to the comic effect of the image. The image captures the viewer’s attention and sparks a sense of humor due to the unexpected and imaginative representation of a dog dressed like a human.</p>
LLaVA-3.8B	<p>The image is funny because it features a dog dressed in a human-like dress, posing in a pose as if it were a person. The dog appears to be dressed in a dress, which is a humorous and unexpected scene, as dogs are not typically dressed like humans. The dog’s pose and the overall scene create a sense of humor and playfulness, making the image a delightful and entertaining piece.</p>

Table 10: Qualitative analysis of compressed LLaVA models with respect to the image. We feed the compressed LLaVA models the image paired with a prompt to generate a response. The compressed LLaVA models are able to understand the visual inputs and output reasonable texts.