⛰ TRAIN SMALL, INFER LARGE: MEMORY-EFFICIENT LoRA TRAINING FOR LARGE LANGUAGE MODELS

Jun Zhang[1,3]*, **Jue Wang**[1,3], **Huan Li**[1,2]†, **Lidan Shou**[1,2]†, **Ke Chen**[1,2],
Yang You[4], **Guiming Xie**[5], **Xuejian Gong**[5], **and Kunlong Zhou**[5]

[1]The State Key Laboratory of Blockchain and Data Security, Zhejiang University
[2]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
[3]College of Computer Science and Technology, Zhejiang University
[4]Department of Computer Science, National University of Singapore
[5]AI Center, Guangdong OPPO Mobile Telecommunications Corp., Ltd.
`{zj.cs,zjuwangjue,lihuan.cs,should,chenk}@zju.edu.cn`,
`youy@comp.nus.edu.sg, {xieguiming,gongxuejian,zhoukunlong}@oppo.com`

ABSTRACT

Large Language Models (LLMs) have significantly advanced natural language processing with exceptional task generalization capabilities. Low-Rank Adaption (LoRA) offers a cost-effective fine-tuning solution, freezing the original model parameters and training only lightweight, low-rank adapter matrices. However, the memory footprint of LoRA is largely dominated by the original model parameters. To mitigate this, we propose LoRAM, a memory-efficient LoRA training scheme founded on the intuition that many neurons in over-parameterized LLMs have low training utility but are essential for inference. LoRAM presents a unique twist: it trains on a pruned (small) model to obtain pruned low-rank matrices, which are then recovered and utilized with the original (large) model for inference. Additionally, minimal-cost continual pre-training, performed by the model publishers in advance, aligns the knowledge discrepancy between pruned and original models. Our extensive experiments demonstrate the efficacy of LoRAM across various pruning strategies and downstream tasks. For a model with 70 billion parameters, LoRAM enables training on a GPU with only 20G HBM, replacing an A100-80G GPU for LoRA training and 15 GPUs for full fine-tuning. Specifically, QLoRAM implemented by structured pruning combined with 4-bit quantization, for LLaMA-3.1-70B (LLaMA-2-70B), reduces the parameter storage cost that dominates the memory usage in low-rank matrix training by 15.81× (16.95×), while achieving dominant performance gains over both the original LLaMA-3.1-70B (LLaMA-2-70B) and LoRA-trained LLaMA-3.1-8B (LLaMA-2-13B). Code is available at **`https://github.com/junzhang-zj/LoRAM`**.

1 INTRODUCTION

Large language models (LLMs), such as GPT-4 (OpenAI, 2023), LLaMA (Touvron et al., 2023a;b; Dubey et al., 2024), and PaLM (Chowdhery et al., 2023), have recently revolutionized natural language applications. These models excel in task generalization, driven by their exponential increase in scale, with some exceeding 400 billion parameters (Dubey et al., 2024). Fine-tuning pre-trained LLMs is critical for task-specific customization, enhancing desired behaviors while mitigating undesired ones (Qi et al., 2024). However, this process is constrained by substantial memory requirements; for instance, fine-tuning a 70B LLaMA in 16-bit precision demands over 1178GB[1] of memory, necessitating an expensive setup of 15 GPUs (A100 or H100, each with 80GB HBM).


* Work done during an internship at OPPO AI Center.
† Huan Li and Lidan Shou are the corresponding authors.
[1]Training is performed on one sample with a length of 4K using `BF16` mixed precision with the Adam optimizer, incorporating gradient checkpointing.

To mitigate the high cost of fine-tuning LLMs, parameter-efficient fine-tuning (Li & Liang, 2021; Lester et al., 2021; Liu et al., 2021; Qiu et al., 2023; Liu et al., 2024b; 2022), particularly Low-Rank Adaption (LoRA) (Hu et al., 2022) and its variants (Liu et al., 2024a; Ding et al., 2023; Zi et al., 2023; Zhang et al., 2023b; Kalajdzievski, 2023), freezes the original LLM weights and only updates the injected low-rank matrices to adapt to new tasks under limited resources. However, during training, they still struggle with the significant memory footprint of the parameters of the base LLM, even with quantization (Dettmers et al., 2023; Xu et al., 2024; Li et al., 2024; Guo et al., 2024; Frantar et al., 2023; Chai et al., 2023). Typically, they reduce the precision to 4 bits at most due to quality considerations. This memory dilemma raises an interesting and necessary question:

Can we further reduce the memory overhead of the base model during LoRA training while still maintaining the inference accuracy?

Our answer is a resounding *Yes*! In this paper, we propose **M**emory-efficient **LoRA** training, coined LoRAM, a novel scheme to reduce the memory overhead of LoRA fine-tuning for LLMs. We revisit the training and inference process of the LoRA paradigm, building on it with a unique twist: Unlike typical LoRA, which uses the same original model for training and inference, LoRAM employs different models at each stage, i.e., it trains a *pruned (small)* model by updating the pruned LoRA weights and then performs inference on the *original (large)* model with the recovered low-rank matrices. This recovery process reshapes the pruned matrices to ensure be merged into the original model, allowing for the updating of unpruned weights while utilizing the pruned weights during inference. The key insight driving our approach comes from reconciling two seemingly contradictory concepts. The scaling laws (Hoffmann et al., 2022; Kaplan et al., 2020; Hernandez et al., 2021) suggest that a large number of parameters of LLMs is essential for effective model generalization. Conversely, sparsity in LLMs (Zhang et al., 2024b; Ma et al., 2023; Sun et al., 2024; Frantar & Alistarh, 2023a; Xia et al., 2024) show that these pre-trained models can be compressed by removing redundant weights. The goal is to minimize the difference in the model's output before and after pruning. However, such methods tend to falter at higher pruning ratios and aggressively pruned models lose critical reasoning capabilities, e.g., only pruning 10%~20% (Ma et al., 2023; Frantar & Alistarh, 2023a). Our intuition builds on this: some critical parameters contribute significantly to fine-tuning, while other parameters, though essential for inference, usually remain unchanged during fine-tuning. Therefore, LoRAM leverages this insight by updating the weights retained through pruning from LoRA training (yellow blocks in Fig. 1) to significantly reduce memory usage and training time, while employing the pruned weights (blue blocks in Fig. 1) during inference to enhance generation performance (see Section 3.5).
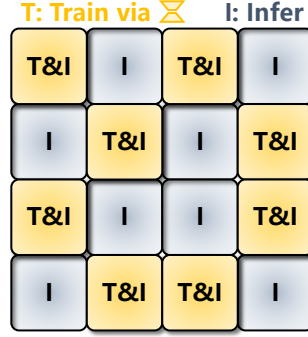
Figure 1: Idea of LoRAM

Despite the significant reduction in memory cost achieved by the pruning-recovery process of LoRAM, maintaining gains at more aggressive pruning rates (e.g., 65% or higher) remains challenging. We attribute this to the knowledge inconsistency between the pruned model used for training and the original model used for inference. To address this, we propose an effective alignment strategy: low-cost continual pre-training of the pruned model on a small dataset. This alignment is performed once offline, allowing the model's publisher to execute it. For instance, Meta AI could release a set of aligned pruned models for LLaMA-3, enabling low-resource users to fine-tune large models for customized tasks using LoRAM. Notably, as a bonus, LoRAM seamlessly integrates with existing quantization schemes designed for LoRA, such as QLoRA, forming QLoRAM, which further reduces memory overhead.

The contributions of this work are summarized as follows:

(1) *Novel Training Scheme*: We propose LoRAM, a memory-efficient LoRA training scheme. LoRAM trains a pruned model by updating the pruned low-rank matrices and then uses dimensionally recovered low-rank matrices to integrate with the original model for inference. The process significantly reduces the memory consumption incurred by the model parameters during training, and synergistically boosts performance by leveraging the full original parameters during inference. Thus, LoRAM efficiently enhances performance under limited device memory resources.

(2) *Effective Alignment Strategy*: We identify that the knowledge inconsistency between the pruned model used for training and the original model used for inference limits the performance gain of LORAM under aggressive pruning rates. To tackle this, we train the pruned model on a small amount of general corpus to achieve alignment, which is a one-shot offline process and can be easily performed by the model publisher.

(3) *Extensive Experimental Evaluation*: We conduct comprehensive experiments to validate the effectiveness of LORAM across various pruning algorithms, models of different sizes, and tasks in different domains. Notably, QLORAM which combines LORAM with structured pruning and 4-bit quantization reduces the memory cost of LLaMA-2-70B parameters by $16.95\times$ while effectively achieving performance gains superior to both the original LLaMA-2-70B and LLaMA-2-13B fine-tuned with LoRA.

2 MEMORY-EFFICIENT LoRA TRAINING — LoRAM

2.1 LOW-RANK ADAPTATION

Given a pre-trained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{m \times n}$, a typical full-parameter fine-tuning (Li et al., 2023b) process adapts to new tasks by updating the entire full-rank matrix $\mathbf{W}_0$. Inspired by the insight that pre-trained weights of LLMs exhibit low "intrinsic dimension" when adapting to specific tasks (Aghajanyan et al., 2021), LoRA (Hu et al., 2022) further suggests that the updated weights have a low "intrinsic rank". Consequently, LoRA reparameterizes the model weights as $\mathbf{W}_0 + \mathbf{W}_\Delta = \mathbf{W}_0 + \mathbf{BA}$, where $\mathbf{B} \in \mathbb{R}^{m \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times n}$, and $\mathbf{W}_\Delta = \mathbf{BA}$ represents a low-rank decomposition matrix with the rank $r \ll \min(m, n)$.

During training, as illustrated in Fig. 2 (a), the pre-trained weight matrix $\mathbf{W}_0$ is frozen to avoid gradient computation. Instead, the low-rank matrices $\mathbf{B}$ and $\mathbf{A}$ are updated to enable parameter-efficient fine-tuning, which defaults to standard supervised fine-tuning, with the objective function $\mathcal{L}_{\text{SFT}}$ defined as the cross-entropy loss between the predicted logits and the ground-truth answers. Given an input feature vector $\mathbf{x}$ of length m, the forward pass of LoRA modifies the output activation from fully fine-tuning, represented by $\mathbf{h} = \mathbf{x}\mathbf{W}_0$ (of length n), to:

$$\mathbf{h} = \mathbf{x}\mathbf{W}_0 + \mathbf{x}\mathbf{W}_\Delta = \mathbf{x}\mathbf{W}_0 + \mathbf{x}\mathbf{BA}. \tag{1}$$

Once low-rank matrices $\mathbf{B}^\star$ and $\mathbf{A}^\star$ are trained by minimizing the $\mathcal{L}_{\text{SFT}}$, as shown in the Fig. 2 (c), the computation of activation $\mathbf{h}$ for $\mathbf{x}$ is reformulated to improve inference efficiency:

$$\mathbf{h} = \mathbf{x}(\mathbf{W}_0 + \mathbf{W}_\Delta^\star) = \mathbf{x}(\mathbf{W}_0 + \mathbf{B}^\star \mathbf{A}^\star). \tag{2}$$

2.2 MEMORY-EFFICIENT LoRA TRAINING

Consider the LLaMA-2-13B model, we introduce low-rank matrices ($r = 8$) for the four projection matrices ($\mathbf{W}_\text{q}$, $\mathbf{W}_\text{k}$, $\mathbf{W}_\text{v}$, and $\mathbf{W}_\text{o}$) in the attention layer, the three projection matrices ($\mathbf{W}_\text{up}$, $\mathbf{W}_\text{gate}$, and $\mathbf{W}_\text{down}$) in the MLP layer, and the weight matrix $\mathbf{W}_{\text{lm\_head}}$ in output layer. Despite the additional 32 million parameters, the number of trainable parameters is reduced by $406\times$ compared to the full parameters. Many LoRA variats (Zhou et al., 2024; Zhang et al., 2023a; Kopiczko et al., 2024; Azizi et al., 2024; Wang et al., 2024) aim to address the significant memory overhead associated with $\mathbf{W}_\Delta$ as $\mathbf{W}_0$ scales, but they still necessitate storing a complete copy of $\mathbf{W}_0$ in memory, which dominates training memory usage. Even with quantization methods designed for LoRA (Dettmers et al., 2023; Xu et al., 2024; Li et al., 2024; Guo et al., 2024; Frantar et al., 2023; Chai et al., 2023), training performance constraints often limit the representation of $\mathbf{W}_0$ to 4 bits. Consequently, for the LLaMA-2-13B, storage requirements are reduced from 26 GB in `FP16` to 6.5GB in `NF4`. However, this is still significantly higher than the memory required for $\mathbf{W}_\Delta$ in `BF16`, which occupies only 64MB of storage and has a peak memory requirement of 576MB during training. Thus, the memory needed for the frozen quantized $\mathbf{W}_0$ is $11.5\times$ greater than that required for learnable $\mathbf{W}_\Delta$.

To mitigate the memory overhead dominated by $\mathbf{W}_0$ while achieving inference performance gain, we propose a memory-efficient LoRA training called LORAM. LORAM first prunes the model to a smaller size and performs LoRA fine-tuning on the pruned model. After training, it recovers the LoRA weights, applies them to the original model, and then conducts inference. We now describe the various stages of LORAM. The complete algorithm of LORAM is presented in Appendix F.
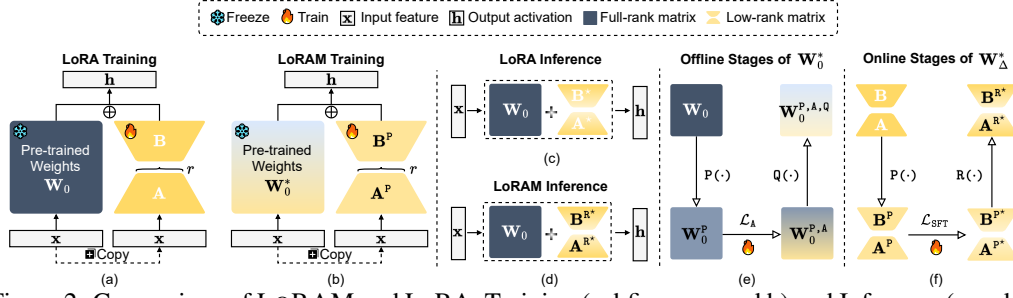
Figure 2: Comparison of LoRAM and LoRA: Training (subfigures a and b) and Inference (c and d). Key stages include the offline process of the frozen full-rank matrix $\mathbf{W}_0^\star$ (subfigure e) and the online generation of the learnable low-rank matrix $\mathbf{W}_\Delta^\star$ (f) during LoRAM training (b) and inference (d).

Pruned Full-Rank Weight Generation. First, we employ a pruning algorithm $\mathtt{P}(\cdot)$ to derive the pruned weight matrix $\mathbf{W}_0^\mathtt{P}$ from the original weights $\mathbf{W}_0$. Specifically, $\mathbf{W}_0^\mathtt{P}$ is computed as:

$$\mathbf{W}_0^\mathtt{P} = \mathtt{P}(\mathbf{W}_0) = \mathbf{W}_0 \circ \mathbf{M}^\mathtt{P}, \tag{3}$$

where $\mathbf{M}^\mathtt{P} \in \{0,1\}^{m \times n}$ is a binary mask matrix indicating retained parameters ('1') and pruned parameters ('0'), and $\circ$ denotes the Hadamard product.

Pruned Low-Rank Matrix Training. After obtaining the pruned weight matrix $\mathbf{W}_0^\mathtt{P}$, we modify the standard LoRA training process. Instead of updating the low-rank matrices $\mathbf{B}$ and $\mathbf{A}$ for the original $\mathbf{W}_0$, we train the pruned low-rank decomposition matrix $\mathbf{W}_\Delta^\mathtt{P} = \mathbf{W}_\Delta \circ \mathbf{M}^\mathtt{P} = \mathbf{B}^\mathtt{P} \mathbf{A}^\mathtt{P}$, while keeping $\mathbf{W}_0^\mathtt{P}$ frozen as shown in Fig. 2 (b). The output activation $\mathbf{h}$ for an input feature vector $\mathbf{x}$ is calculated as:

$$\mathbf{h} = \mathbf{x}\mathbf{W}_0^\mathtt{P} + \mathbf{x}\mathbf{W}_\Delta^\mathtt{P} = \mathbf{x}\mathbf{W}_0^\mathtt{P} + \mathbf{x}(\mathbf{B}^\mathtt{P}\mathbf{A}^\mathtt{P}). \tag{4}$$

Recovered Low-Rank Matrix Generation. By optimizing the objective function $\mathcal{L}_{\text{SFT}}$, we obtain the trained pruned low-rank matrix $\mathbf{W}_\Delta^{\mathtt{P}^\star}$. To fully leverage the original model weights for improved inference performance, we introduce a recovery function $\mathtt{R}(\cdot)$, guided by the pruning mask $\mathbf{M}^\mathtt{P}$. This function recovers the shape of the trained low-rank matrix by filling zeros at pruned positions, resulting in $\mathbf{W}_\Delta^{\mathtt{R}^\star}$ as follows:

$$\mathbf{W}_\Delta^{\mathtt{R}^\star} = \mathbf{B}^{\mathtt{R}^\star}\mathbf{A}^{\mathtt{R}^\star} = \mathtt{R}(\mathbf{W}_\Delta^{\mathtt{P}^\star}) = \mathbf{W}_\Delta^{\mathtt{P}^\star} \circ (1 - \mathbf{M}^\mathtt{P}). \tag{5}$$

This operation ensures that the recovered low-rank matrix $\mathbf{W}_\Delta^{\mathtt{R}^\star}$ can be seamlessly merged with the original pre-trained weights $\mathbf{W}_0$, forming $\mathbf{W}_0 + \mathbf{W}_\Delta^{\mathtt{R}^\star}$ as follows:

$$(\mathbf{W}_0 + \mathbf{W}_\Delta^{\mathtt{R}^\star})[i,j] = \begin{cases} \mathbf{W}_0[i,j] & \text{if } \mathbf{M}^\mathtt{P}[i,j] = 1 \\ \mathbf{W}_0[i,j] + \mathbf{W}_\Delta^{\mathtt{R}^\star}[i,j] & \text{if } \mathbf{M}^\mathtt{P}[i,j] = 0 \end{cases} \tag{6}$$

This formula indicates that, for positions where the pruning mask $\mathbf{M}^\mathtt{P}$ is '1', the merged matrix $\mathbf{W}_0 + \mathbf{W}_\Delta^{\mathtt{R}^\star}$ retains the original values from the pre-trained matrix $\mathbf{W}_0$. For positions where the mask is '0', the elements in the merged matrix are updated to be the sum of the corresponding values from $\mathbf{W}_0$ and the recovered low-rank matrix $\mathbf{W}_\Delta^{\mathtt{R}^\star}$.

Recovered Low-Rank Matrix Inference. Once we obtain the recovered low-rank matrix $\mathbf{W}_\Delta^{\mathtt{R}^\star}$, during inference, the forward pass output activation $\mathbf{h}$ for an input feature $\mathbf{x}$ is computed as follows:

$$\mathbf{h} = \mathbf{x}(\mathbf{W}_0 + \mathbf{W}_\Delta^{\mathtt{R}^\star}) = \mathbf{x}(\mathbf{W}_0 + \mathbf{B}^{\mathtt{R}^\star}\mathbf{A}^{\mathtt{R}^\star}). \tag{7}$$

Our experiments (see Section 3.2 and Section 3.3) show that LoRAM maintains high performance across various pruning strategies $\mathtt{P}(\cdot)$, including structured (Ma et al., 2023) and non-structured pruning (semi-structured & unstructured pruning) (Frantar & Alistarh, 2023a). The four stages outlined above summarize the core steps of LoRAM. To avoid notational clutter, we have streamlined the algorithmic details. Nonetheless, three key considerations for deployment must be emphasized:

C_1 **Pruned Full-Rank Weight Generation:** For non-structured pruning, the matrix dimension remains unchanged, with $\mathbf{W}_0^P$ compressed into a sparse matrix populated by zeros. In structured pruning, weights are physically removed, yielding a compact, dense $\mathbf{W}_0^P$.

C_2 **Recovered Low-Rank Matrix Generation:** For non-structured pruning, the weights in $\mathbf{W}_\Delta^P$ corresponding to pruned positions in $\mathbf{M}^P$ are excluded from backpropagation by setting their gradients to zero, ensuring that only the retained components are updated during training.

C_3 **Recovered Low-Rank Matrix Inference:** For non-structured pruning, the shapes of both the pre-trained and low-rank weights are identical (see C_1), and the gradients of the pruned weights are blocked (see C_2). Consequently, we can bypass the recovery phase, resulting in $\mathbf{W}_\Delta^{R^\star} = \mathbf{B}^{R^\star}\mathbf{A}^{R^\star} = \mathbf{B}^{P^\star}\mathbf{A}^{P^\star}$. In the case of structured pruning, the shapes of the weight matrices vary significantly across different pruning strategies. To simplify the definitions, we standardize the recovery process using the pruning mask.

To clearly illustrate the evolution of weight matrix dimensions across these stages, we take LLM-Pruner (Ma et al., 2023) as an example in Appendix C, visualizing the transformation from $\mathbf{W}_0 \Rightarrow \mathbf{W}_0^P$, $\mathbf{W}_\Delta \Rightarrow \mathbf{W}_\Delta^P$, and $\mathbf{W}_\Delta^{P^\star} \Rightarrow \mathbf{W}_\Delta^{R^\star}$ under LoRAM with structured pruning.

Pruned Full-Rank Weight Alignment. Given the original pre-trained weights $\mathbf{W}_0$, the optimal low-rank matrix learned is $\mathbf{W}_\Delta^\star$. Similarly, for pruned weights $\mathbf{W}_0^P$, the optimal low-rank counterpart is $\mathbf{W}_\Delta^{P^\star}$. If the knowledge encoded in $\mathbf{W}_0$ and $\mathbf{W}_0^P$ is closely aligned, the knowledge embedded in $\mathbf{W}_\Delta^{P^\star}$ should approximate that of $\mathbf{W}_\Delta^\star$. Consequently, the recovered matrix $\mathbf{W}_\Delta^{R^\star}$ should effectively pair with $\mathbf{W}_0$, yielding performance improvements similar to those from $\mathbf{W}_\Delta^\star$. However, the pruning function $P(\cdot)$ disrupts some of the knowledge embedded in the original weights, leading to a mismatch between $\mathbf{W}_0$ and $\mathbf{W}_0^P$. Such knowledge mismatch causes $\mathbf{W}_\Delta^{R^\star}$, when paired with $\mathbf{W}_0$, to deliver suboptimal performance, particularly at higher pruning ratios.

To address the knowledge mismatch, we propose an efficient alignment scheme, namely continual pre-training of pruned weights $\mathbf{W}_0^P$ into $\mathbf{W}_0^{P,A}$ on a small, general corpus $\mathcal{D}_A$. Formally, we minimize the alignment loss $\mathcal{L}_A$ defined as following:

$$\mathcal{L}_A = -\mathbb{E}_{\mathbf{s} \in \mathcal{D}_A} \left[\sum_{t=1}^{|\mathbf{s}|} \log p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_{<t}; \mathbf{W}_0^{P,A}\right) \right], \tag{8}$$

where $p(\mathbf{s}_{t+1} \mid \mathbf{s}_{<t}; \mathbf{W}_0^{P,A})$ represents the model's predict likelihood of generating the next token $\mathbf{s}_{t+1}$ given the first t tokens $\mathbf{s}_{<t}$ of input sequence $\mathbf{s}$ (token number is $|\mathbf{s}|$) and current parameter matrices $\mathbf{W}_0^{P,A}$. This alignment process is a one-time, offline operation on a relatively small corpus (about 105 million tokens), making it a cost-effective solution for model publishers. Alongside the base model, they can release the aligned pruned model, enabling low-resource users to fine-tune the base model via LoRAM for specific downstream tasks.

Pruned Full-Rank Weight Quantization. The design of LoRAM inherently supports the seamless integration of quantization techniques, further reducing memory consumption during training by applying quantization to pruned models. For example, by adapting the LoRA-specific quantization scheme QLoRA (Dettmers et al., 2023), LoRAM extends into QLoRAM, the pruned full-rank weight matrix is quantized into the NF4 format, while the pruned low-rank matrices $\mathbf{B}^P$ and $\mathbf{A}^P$ remain in full or half precision, striking a balance between memory efficiency and fine-tuning quality.

Formally, given a quantization function $Q(\cdot)$, during training, the forward pass output activation vector $\mathbf{h}$ for an input feature vector $\mathbf{x}$ is computed as:

$$\mathbf{h} = \mathbf{x}Q(\mathbf{W}_0^P) + \mathbf{x}\mathbf{B}^P\mathbf{A}^P = \mathbf{x}\mathbf{W}_0^{P,Q} + \mathbf{x}\mathbf{B}^P\mathbf{A}^P, \tag{9}$$

where $\mathbf{W}_0^{P,Q}$ represents the full-rank weight $\mathbf{W}_0$ after undergoing pruning via $P(\cdot)$, followed by quantization using $Q(\cdot)$. We can also quantize the pruned full-rank weight after alignment to $\mathbf{W}_0^{P,A,Q}$. For inference, unless additional quantization is required, QLoRAM operates identically to LoRAM, as shown in Fig. 2 (d). It utilizes the original full-rank weights $\mathbf{W}_0$ alongside the recovered low-rank matrices $\mathbf{B}^{R^\star}$ and $\mathbf{A}^{R^\star}$ to perform the forward pass according to Eq. (7). In summary, for

LoRAM, as shown in Fig. 2 (e), the offline processing path of the frozen full-rank matrix $\mathbf{W}_0^\star$, which minimizes GPU memory usage, is $\mathbf{W}_0 \xrightarrow{\mathtt{P}(\cdot)} \mathbf{W}_0^\mathtt{P} \xrightarrow{\mathcal{L}_\mathtt{A}} \mathbf{W}_0^{\mathtt{P},\mathtt{A}} \xrightarrow{\mathtt{Q}(\cdot)} \mathbf{W}_0^{\mathtt{P},\mathtt{A},\mathtt{Q}}$; Fig. 2 (f) shows that the online generation path for the trained low-rank matrix $\mathbf{W}_\Delta^\star$ is $\mathbf{W}_\Delta \xrightarrow{\mathtt{P}(\cdot)} \mathbf{W}_\Delta^\mathtt{P} \xrightarrow{\mathcal{L}_\mathtt{SFT}} \mathbf{W}_\Delta^{\mathtt{P}\star} \xrightarrow{\mathtt{Q}(\cdot)} \mathbf{W}_\Delta^{\mathtt{R}\star}$.

3 EXPERIMENTS

3.1 SETUP

Pre-train Corpus. To align the inconsistent knowledge between the pruned model during training and the original model during inference, we apply LoRAM to continual pre-training LLMs on a mixed corpus of FineWeb (Penedo et al., 2024) and OpenWebMath (Paster et al., 2023). Notably, this alignment process is a one-time, offline operation that can be executed by model publishers.

Fine-tuning Data. Following the fine-tuning setup of LoRA (Hu et al., 2022), we primarily conduct supervised fine-tuning (SFT) on the OpenHermes-2.5 (Teknium, 2023) (referred to as Open-Hermes) and OpenOrca (Lian et al., 2023) datasets. To effectively assess the overall fine-tuning performance, we evaluate test perplexity not only on in-domain test sets constructed from the instruction fine-tuning data but also on out-of-domain test sets built from Alpaca (Taori et al., 2023).

Downstream Task. We focus on the performance of LoRAM in various downstream tasks, including MathQA (Amini et al., 2019) and GSM8K (Cobbe et al., 2021) in mathematical reasoning, six tasks—Arc Challenge & Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Open-BookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021) in common sense reasoning, and HumanEval (Chen et al., 2021) in code generation.

Sparsification & Quantization. For sparsification $\mathtt{P}(\cdot)$, we first establish a variant LoRAM-RAND randomly structured pruning and adapt LoRAM to another three variants based on leading approaches: LoRAM-STRU with the structured pruning LLM-Pruner[2] (Ma et al., 2023) and LoRAM-SEMI and LoRAM-UNST with the non-structured (semi-structured & unstructured) pruning SparseGPT[3] (Frantar & Alistarh, 2023a). For quantization $\mathtt{Q}(\cdot)$, we achieve QLoRAM by combining LoRAM with the LoRA-tailored quantization algorithm QLoRA (Dettmers et al., 2023). The storage cost of the original model primarily drives the memory consumption during LoRA weights training. Thus, we define the *parameter reduction ratio* as the count of parameters before and after pruning, to evaluate the memory efficiency of baselines. The details of our experiment setups and hyperparameters are provided in Appendix B.

3.2 FINE-TUNING CONVERGENCE

We investigate the convergence trends of LoRAM across varying model scales (LLaMA-2-13B & LLaMA-2-70B) and different instruction-tuning datasets (OpenHermes & OpenOrca). To assess training performance, we track perplexity over training iterations on both out-of-domain (Alpaca) and in-domain (OpenHermes or OpenOrca) test sets, as shown in Fig. 3 and Fig. 4.

Out-of-Domain Performance. LoRAM consistently achieves out-of-domain performance with similar trends, positioned between the LoRA fine-tuned models of the same scale and smaller models, across different models and datasets. As shown in Figs. 3 and 4 (a), for the 13B model, the perplexity of LoRAM variants pruned by different algorithms is lower than that of the LoRA-trained 7B model but higher than the LoRA-trained 13B model, with LoRAM-RAND and LoRAM-STRU achieving a 2.17× parameter reduction. Similarly, as shown Figs. 3 and 4 (c), for the 70B model, this reduction extends to 12.84× under similar convergence trends.

In-Domain Performance. LoRAM shows limited improvement in in-domain performance, likely due to overfitting when the base models are fine-tuned with LoRA, resulting in relatively lower

---

[2] https://github.com/horseee/LLM-Pruner (Apache-2.0 license)
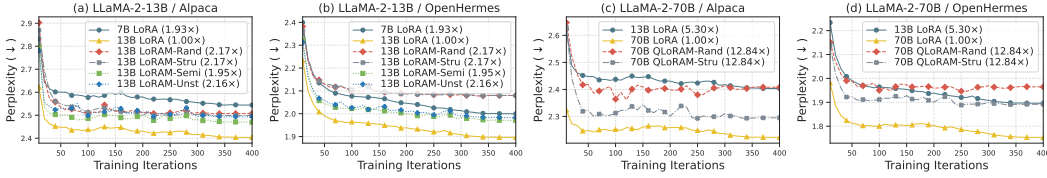[3] https://github.com/IST-DASLab/sparsegpt (Apache-2.0 license)

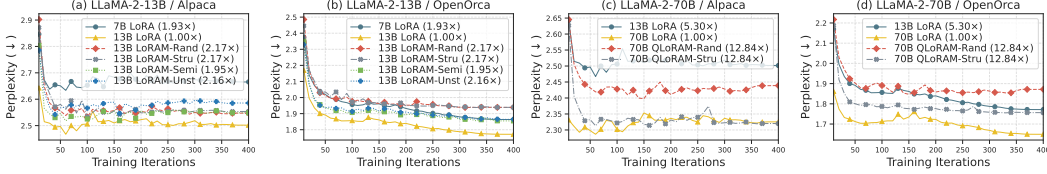Figure 3: The test perplexity of training LLaMA-2-13B & LLaMA-2-70B on OpenHermes.

Figure 4: The test perplexity of training LLaMA-2-13B & LLaMA-2-70B on OpenOrca.

perplexity. This is further supported by downstream evaluations, where models that excel in in-domain perplexity often underperform in downstream tasks. As shown in Figs. 3 and 4 (b), while the LoRA-trained 7B model outperforms 13B LoRAM-RAND and LoRAM-STRU on in-domain tests, it underperforms on several downstream tasks as shown in Section 3.3.

Non-Structured LoRAM Excels in In-Domain. The non-structured variants (LoRAM-SEMI & LoRAM-UNST) consistently outperform their structured counterparts (LoRAM-RAND & LoRAM-STRU) on in-domain test sets. As shown in Fig. 3 (a) vs. Fig. 3 (b) and Fig. 4 (a) vs. Fig. 4 (b), the in-domain perplexity of LoRAM-SEMI and LoRAM-UNST is notably lower, while their out-of-domain performance shows less pronounced differences. This advantage likely arises from the more selective weight pruning in the non-structured variants, which preserves information capture capabilities similar to the original model, thus enhancing in-domain performance.

Non-Random LoRAM Benefits from Scaling. The performance gains of the non-random LoRAM become more evident as the model size grows. As shown in (a,b) of Figs. 3 and 4 vs. (c,d) of Figs. 3 and 4, LoRAM-STRU outperforms LoRAM-RAND considerably on the 70B model, while the difference is marginal on the 13B model. This indicates that larger models exhibit greater differences in the redundancy of individual weights, making selective pruning more effective[4].

3.3 DOWNSTREAM TASK PERFORMANCE

We evaluate the performance of various models trained with LoRAM on different instruction data across three downstream tasks: mathematical reasoning, common sense reasoning (CSR), and code generation. Results are summarized in Tables 1 to 3. We highlight the *core competition scenario* with a gray background, which includes the untrained original model and a smaller sibling model trained with LoRA. For instance, for LoRAM-trained LLaMA-2-13B, we report the scores of the 13B without fine-tuning and the LoRA-trained 7B model. Blue backgrounds of varying intensity indicate the degree of improvement for each LoRAM variant relative to the *core competition scenario*: darker shades indicate greater improvements, while lighter shades signify smaller gains.

Overall, we observe that most LoRAM variants outperform the core competitive baseline across all downstream tasks, particularly in mathematical and common sense reasoning. This improvement is further amplified by increasing the model scale. Specifically, as shown in Table 1, the 70B LoRAM-RAND and LoRAM-STRU models achieve a 12.84× reduction in parameters compared to the original 70B model (70B w/o FT), exceeding the 5.30× reduction of the LoRA-trained 13B model. In terms of performance, LoRAM improves the original 70B model's score on GSM8K from 52% to 57%, significantly outperforming the LoRA-trained 13B model, which only achieved 37%. These results demonstrate that updating low-rank matrices on pruned models effectively re-

---

[4]The trained low-rank matrices are visualized in Appendix D, and the update patterns they exhibit somewhat align with these insights.

Table 1: Accuracy (%) of the MathQA (1-shot) & GSM8K (8-shots) in the mathematical domain under LLaMA-2. ▲ indicates the theoretical parameters reduction of non-structured pruning. However, these parameters are filled with zeros in actual training, so the memory footprint is not reduced.

METHOD	OPENHERMES		OPENORCA		PARAMETER
	MATHQA	GSM8K	MATHQA	GSM8K	REDU. RATIO
13B w/o FT	32.60	24.26	32.93	23.35	1.00×
7B LORA	29.61	22.82	30.95	13.87	1.93×
13B LORAM-RAND	33.77	27.22	32.83	25.93	2.17×
13B LORAM-STRU	33.80	24.64	33.07	24.49	2.17×
13B LORAM-SEMI	31.76	36.92	33.07	27.29	▲ 1.95×
13B LORAM-UNST	30.12	31.92	32.70	26.61	▲ 2.16×
70B w/o FT	39.53	52.01	39.53	52.01	1.00×
13B LORA	32.03	36.69	33.63	25.70	5.30×
70B QLORAM-RAND	39.66	57.62	39.40	55.72	12.84×
70B QLORAM-STRU	39.77	57.16	39.73	54.44	12.84×

Table 2: Average accuracy (%) of the CSR in the common sense reasoning domain (1-shot) under the LLaMA-2. Baseline results for each subtask of CSR are detailed in Appendix E.

METHOD	OPENHERMES	OPENORCA	PARAMETER
	MEAN ± STD	MEAN ± STD	REDU. RATIO
13B w/o FT	64.28±1.30	64.28±1.30	1.00×
7B LORA	61.51±1.29	61.42±1.30	1.93×
13B LORAM-RAND	64.64±1.29	64.49±1.30	2.17×
13B LORAM-STRU	64.42±1.29	64.32±1.29	2.17×
13B LORAM-SEMI	64.38±1.29	64.73±1.30	▲ 1.95×
13B LORAM-UNST	64.12±1.29	64.68±1.29	▲ 2.16×
70B w/o FT	68.69±1.27	68.69±1.27	1.00×
13B LORA	65.05±1.29	65.40±1.29	5.30×
70B QLORAM-RAND	68.99±1.27	68.46±1.27	12.84×
70B QLORAM-STRU	69.10±1.27	68.94±1.27	12.84×

Table 3: PASS@1(%) and PASS@10(%) of HumanEval in the code generation domain under LLaMA-2. The best results for all baselines are reported, selected from TEMPERATURE settings in {0.0, 0.2, 0.4, 0.6, 0.8} with TOP_P fixed at 0.95.

METHOD	OPENHERMES		OPENORCA		PARAMETER
	PASS@1	PASS@10	PASS@1	PASS@10	REDU. RATIO
13B w/o FT	17.68	35.37	17.68	35.37	1.00×
7B LORA	15.24	28.04	15.85	26.21	1.93×
13B LORAM-RAND	19.51	33.54	19.51	32.32	2.17×
13B LORAM-STRU	17.68	35.37	17.07	31.71	2.17×
13B LORAM-SEMI	20.12	35.37	18.29	39.63	▲ 1.95×
13B LORAM-UNST	22.56	34.15	18.29	37.20	▲ 2.16×
70B w/o FT	31.71	58.54	31.71	58.54	1.00×
13B LORA	18.29	35.98	18.29	39.02	5.30×
70B QLORAM-RAND	29.27	57.32	31.71	56.71	12.84×
70B QLORAM-STRU	32.32	58.54	32.32	59.15	12.84×

duces memory requirements during training. Merging the recovered low-rank matrices into the original model yields substantial performance gains during inference.

3.4 ADAPTION TO LLaMA-3.1

Here, we extend LoRAM-Stru to LLaMA-3.1 herds and investigate two key questions: (1) How does LoRAM perform in terms of perplexity and downstream tasks within this model series? (2) What is the effect of continued pre-training iteration steps (proportional to corpus size) on performance?

As shown in Fig. 5 (a,b), QLORAM-STRU for the LLaMA-3.1-70B model exhibits consistent trends across both out-of-domain and in-domain test sets. It achieves a 15.81× parameters reduction while its perplexity falls between that of the smaller LoRA-trained 8B and LoRA-trained 70B.
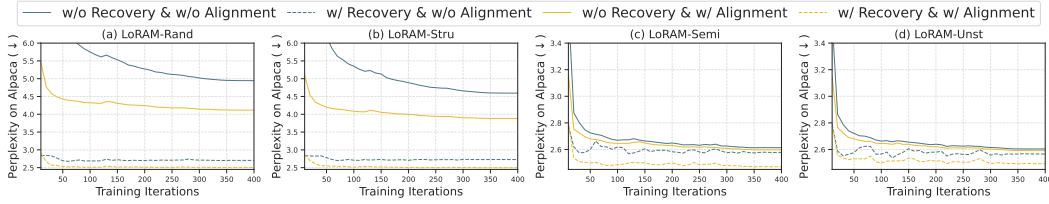
Figure 6: Necessity of Recovery & Alignment across different pruning strategies on LLaMA-2-13B.

In downstream tasks (Fig. 5 (c)), QLORAM-STRU significantly exceeds the 8B w/o FT, 8B LoRA, and 70B w/o FT, even surpassing the LoRA-trained 70B on MathQA and HumanEval (Pass@10). Moreover, we observe that a minimal pre-training corpus can yield substantial performance gains. For instance, QLORAM-STRU 200, with just 200 updates (about 13 million tokens), achieves a 15.81× parame-

Figure 5: The test perplexity & downstream performance of training LLaMA-3.1-70B on OpenHermes.

ter reduction alongside performance improvements. This one-time, low-cost alignment allows publishers to offer aligned pruned models for low-resource users to customize tasks.
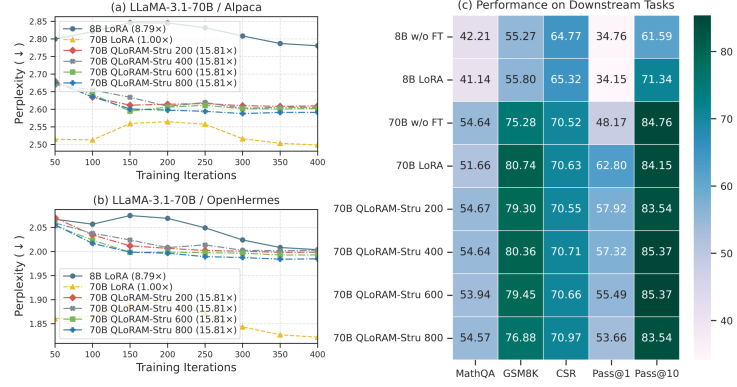
3.5 NECESSITY OF RECOVERY & ALIGNMENT

We conduct an ablation study on two critical phases of LORAM: recovery and alignment. To assess their necessity, we analyze the convergence trends of various pruning variants on the Alpaca test set using LLaMA-2-13B.

Impact of Recovery. we compare the standard approach with an alternative setup where the pruned low-rank matrices are directly combined with the pruned full-rank model weights (w/o Recovery) and track perplexity changes over iterations. As shown in Fig. 6, for all four pruning strategies, models without the recovery phase (solid lines, w/o Recovery & *) consistently exhibit higher perplexity compared to those with recovery (dashed lines, w/ Recovery & *), particularly in structured LORAM (see in Fig. 6 (a) and (b)). This highlights that the recovery phase leverages relatively redundant neurons during training to enhance inference performance significantly.

Impact of Alignment. We also introduce a variant of LORAM without continual pre-training for alignment (w/o Alignment). As shown in Fig. 6, aligned pruned models (yellow lines, * & w/ Alignment) consistently achieve lower perplexity than unaligned counterparts (blue lines, * & w/o Alignment), irrespective of the pruning strategy or recovery phase. This highlights that even low-cost continual pre-training on a small general corpus effectively narrows the knowledge gap between pruned and original models, enhancing the overall performance of LORAM.

3.6 SCALING LAWS FOR PARAMETER REDUCTION ON LORAM

We explore the impact of scaling the parameter reduction ratios in Fig. 7. The LoRA-trained LLaMA-2-13B (triangles) achieves a 5.30× parameter reduction, while QLORAM-STRU maintains superior perplexity on the Alpaca and further reduces parameters across both instruction datasets. In contrast, naive pruning leads to a significant increase in perplexity with minimal pruning. When the parameter reduction ratio reaches 28.56×, QLORAM-STRU sustains an effective perplexity of approximately 2.5, whereas naive pruning escalates to 621.98. These highlight LORAM's ability to

drastically reduce memory of the base model by updating LoRA weights in the pruned model, while seamlessly integrating with the full model to preserve inference performance.

We then evaluate the performance of models trained with LoRAM on OpenHermes across various downstream tasks under different pruning ratios. As shown in Fig. 8, overall performance improves as the parameter reduction ratio increases from $9.82\times$ to $16.95\times$, before declining. Notably, tasks achieve optimal performance between parameter reduction ratios of $12.84\times$ and $16.95\times$, consistently outperforming 13B LoRA and 70B w/o FT. However, at a parameter reduction ratio of $9.82\times$, despite the larger memory capacity available for LoRAM, downstream performance does not always exceed that of higher parameter reduction ratios. We attribute this to the fact

Figure 7: Effect of scaling parameter reduction ratio.

that lower parameter reduction ratios fine-tune more parameters, potentially degrading the pretrained model's performance on certain tasks (e.g., Fig. 8 (a,c,e)). This effect is also reflected in MathQA, where a fully fine-tuned LoRA model underperforms the pre-trained model without finetuning (see Fig. 8 (b)). Moreover, excessive pruning at a ratio of $28.56\times$ leaves too few neurons to capture the rich information needed for downstream improvements, particularly in tasks like code generation (see Fig. 8 (d,e)).
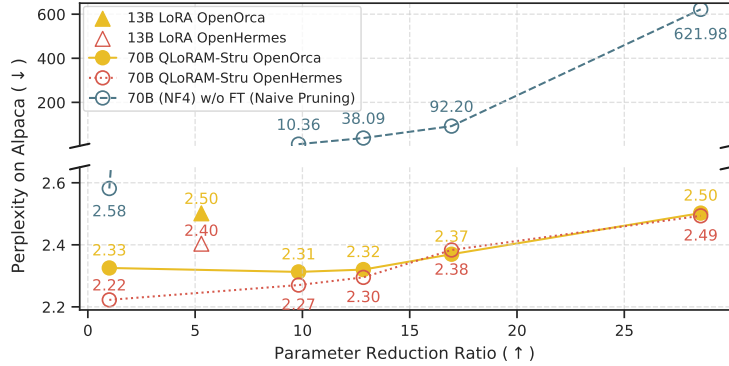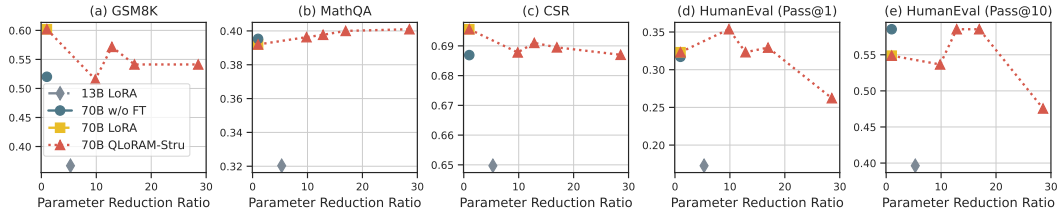

Figure 8: Performance of downstream tasks across different parameter reduction ratios.

4 CONCLUSION

We propose LoRAM, a memory-efficient LoRA training scheme for large language models. LoRAM significantly reduces the count of parameters of the original model by $16.95\times$, while maintaining the performance of large-scale LLM fine-tuning. We identify several open questions for LoRAM, including the potential for reduced inference costs through context-aware computational graph recovery and its applicability to models like vision transformers (Dosovitskiy et al., 2021) and diffusion models (Ho et al., 2020). We hope our work inspires further research on memory-efficient LoRA training from a sparsity perspective and believe LoRAM will serve as a valuable tool for the community, enabling LoRA training of large-scale models on consumer-grade hardware.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 7319–7328. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.568. URL https://doi.org/10.18653/v1/2021.acl-long.568.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *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)*, pp. 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245.

Seyedarmin Azizi, Souvik Kundu, and Massoud Pedram. Lamda: Large model fine-tuning via spectrally decomposed low-dimensional adaptation, 2024. URL https://arxiv.org/abs/2406.12832.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *ICLR 2024 Workshop: How Far Are We From AGI*, 2024. URL https://openreview.net/forum?id=v9L38gCohh.

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. *arXiv preprint arXiv:2212.09095*, 2022.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Yuji Chai, John Gkountouras, Glenn G. Ko, David Brooks, and Gu-Yeon Wei. Int2.1: Towards fine-tunable quantized large language models with error correction through low-rank adaptation, 2023. URL https://arxiv.org/abs/2306.08162.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Tianyi Chen, Tianyu Ding, Badal Yadav, Ilya Zharkov, and Luming Liang. Lorashear: Efficient large language model structured pruning and knowledge recovery, 2023. URL https://arxiv.org/abs/2310.18356.

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4794–4802, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin

Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023. URL http://jmlr.org/papers/v24/22-1144.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023a. URL https://proceedings.mlr.press/v202/frantar23a.html.

Elias Frantar and Dan Alistarh. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023b.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Naibin Gu, Peng Fu, Xiyu Liu, Bowen Shen, Zheng Lin, and Weiping Wang. Light-PEFT: Lightening parameter-efficient fine-tuning via early pruning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7528–7541, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.447. URL https://aclanthology.org/2024.findings-acl.447/.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Han Guo, Philip Greengard, Eric Xing, and Yoon Kim. LQ-loRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xw29VvOMmU.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4340–4349, 2019.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *CoRR*, abs/2102.01293, 2021. URL https://arxiv.org/abs/2102.01293.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241):1–124, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex 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. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 8003–8017. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.507. URL https://doi.org/10.18653/v1/2023.findings-acl.507.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora, 2023. URL https://arxiv.org/abs/2312.03732.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNfLdxr3A.

Sumith Kulal, Panupong Pasupat, Kartik Chandra, Mina Lee, Oded Padon, Alex Aiken, and Percy S Liang. Spoc: Search-based pseudocode to code. In *Advances in Neural Information Processing Systems*, 2019.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 3045–3059. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN. 243. URL https://doi.org/10.18653/v1/2021.emnlp-main.243.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large scale language model society, 2023a.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.353. URL https://doi.org/10.18653/v1/2021.acl-long.353.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LzPWWPAdY4.

Zhikai Li, Xiaoxuan Liu, Banghua Zhu, Zhen Dong, Qingyi Gu, and Kurt Keutzer. Qft: Quantized full-parameter tuning of llms with affordable resources. *arXiv preprint arXiv:2310.07147*, 2023b.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/Open-Orca/OpenOrca, 2023.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *ACL*, 2017.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/0cde695b83bd186c1fd456302888454c-Abstract-Conference.html.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024a.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *ICLR*, 2024b.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021. URL https://arxiv.org/abs/2103.10385.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=J8Ajf9WfXP.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1325–1334, 2019.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Gunho Park, Baeseong Park, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023. URL https://arxiv.org/abs/2310.06786.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hTEGyKf0dZ.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/faacb7a4827b4d51e201666b93ab5fa7-Abstract-Conference.html.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, aug 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*, 2019.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/arXiv.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and finetuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Sheng Wang, Boyang Xue, Jiacheng Ye, Jiyue Jiang, Liheng Chen, Lingpeng Kong, and Chuan Wu. Prolora: Partial rotation empowers more parameter-efficient lora. In *ACL*, 2024.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=09iOdaeOzp.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*, 2022.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *ICLR*, 2024.

Suorong Yang, Peng Ye, Wanli Ouyang, Dongzhan Zhou, and Furao Shen. A clip-powered framework for robust and generalizable data selection. *arXiv preprint arXiv:2410.11215*, 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. *arXiv preprint arXiv:2206.01861*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10. 18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12360–12371, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning, 2023a. URL https://arxiv.org/abs/2308.03303.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 3013–3026, Bangkok, Thailand and virtual meeting, August 2024a. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.178.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/forum?id=lq62uWRJjiY.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse LLMs. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=1ndDmZdT4g.

Jiawei Zhao, Florian Schäfer, and Anima Anandkumar. Zero initialization: Initializing neural networks with only zeros and ones. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=1AxQpKmiTc.

Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pp. 7543–7552. PMLR, 2019.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, and Tiejun Zhao. Lora-drop: Efficient lora parameter pruning based on output evaluation. *CoRR*, abs/2402.07721, 2024. doi: 10.48550/ARXIV.2402.07721. URL https://doi.org/10.48550/arXiv.2402.07721.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices, 2023. URL https://arxiv.org/abs/2309.02411.

APPENDIX

A   RELATED WORK

Low-Rank Adaptation.   LoRA (Low-Rank Adaptation) (Hu et al., 2022) has emerged as a prominent technique for parameter-efficient fine-tuning (PEFT) (Li & Liang, 2021; Lester et al., 2021; Liu et al., 2021; Qiu et al., 2023; Liu et al., 2024b; 2022). By injecting lightweight, trainable low-rank decomposition matrices into frozen pre-trained weights, LoRA enables efficient task customization, especially in resource-constrained settings. Some LoRA variants (Liu et al., 2024a; Ding et al., 2023; Zi et al., 2023; Zhang et al., 2023b; Kalajdzievski, 2023) have been developed to enhance its generalization and robustness, while others (Zhou et al., 2024; Zhang et al., 2023a; Kopiczko et al., 2024; Azizi et al., 2024; Wang et al., 2024; Yang et al., 2024) address the increased memory overhead associated with scaling up model sizes. However, during training, these efficient LoRA variants still struggle with the substantial memory footprint of the original LLM parameters.

LoRA-related Compression.   Model compression techniques like quantization (Han et al., 2015; Jacob et al., 2018; Nagel et al., 2019; Zhao et al., 2019; Yao et al., 2022; Park et al., 2022; Dettmers et al., 2022; Xiao et al., 2022; Frantar et al., 2022), sparsification (Molchanov et al., 2016; Liu et al., 2018; He et al., 2019; Hoefler et al., 2021; Frantar & Alistarh, 2023b; Liu et al., 2023; Bansal et al., 2022), and distillation (Hinton et al., 2015; Cho & Hariharan, 2019; Tang et al., 2019; Touvron et al., 2021; Hsieh et al., 2023; Gu et al., 2024b) have proven effective in reducing the memory footprint of LLM during training and inference. Naturally, the concept of compression has been adapted to LoRA to alleviate the substantial memory consumption dominated by pre-trained model parameters. In particular, LoRA-related quantization schemes (Dettmers et al., 2023; Xu et al., 2024; Li et al., 2024; Guo et al., 2024; Frantar et al., 2023; Chai et al., 2023) have been widely explored, but they still face the limitations of 1-bit precision, typically quantize weights to 4-bit to balance training efficiency with performance. Our work aims to push the boundaries of memory-efficient LoRA training by leveraging sparsification to achieve cost-effective performance improvements. Notably, existing LoRA-related sparsification works (Chen et al., 2023; Zhang et al., 2024a) focus on designing pruning algorithms to slim down models and use LoRA to recover the knowledge of pruned models, thereby producing compact but high-quality models. In contrast, LoRAM enables effective general pruning under high base-model sparsity, whereas (Gu et al., 2024a) focuses on task-specific LoRA sparsification with limited impact on base model memory reduction.

B   EXPERIMENTAL DETAILS

Pre-train Corpus.   To align the inconsistent knowledge between the pruned model during training and the original model during inference, we apply LoRAM to continual pre-training LLMs in a teacher-forcing manner (Bachmann & Nagarajan, 2024) on a mixed corpus of FineWeb (Penedo et al., 2024) and OpenWebMath (Paster et al., 2023). FineWeb, containing over 15TB of cleaned and deduplicated English web data from Common Crawl. OpenWebMath, extracted from over 200 billion HTML files on Common Crawl, provides high-quality mathematical text. Mixing these datasets enhances the pruned model's capabilities in both general and mathematical domains.

Unless specified otherwise, we randomly sample 102,400 instances from both FineWeb and OpenWebMath to construct a mixed dataset with a sequence length of 512, yielding approximately 105 million tokens. The default training batch size is 128, allowing up to 1,600 update steps. We train without data repetition over a sufficiently large corpus to simulate a realistic pre-training scenario. Notably, this alignment process is a one-time, offline operation that model publishers can execute.

Fine-tuning Data.   Following the fine-tuning scenario of LoRA (Hu et al., 2022), we primarily conduct supervised fine-tuning (SFT) on the OpenHermes-2.5 (Teknium, 2023) (referred to as OpenHermes). OpenHermes is a large-scale dataset constructed from synthetically generated instructions and chat samples, encompassing diverse sources such as Airoboros 2.2 (Wang et al., 2023), CamelAI Domain Expert Dataset (Li et al., 2023a), ChatBot Arena (GPT-4 Only) (Zheng et al., 2023), and more. To further demonstrate the general effectiveness of the LoRAM alignment process, we also evaluate LoRAM on the OpenOrca (Lian et al., 2023) dataset. OpenOrca is a widely used

instruction fine-tuning dataset where each data instance represents entries from the FLAN collection (Longpre et al., 2023), augmented by submitting the listed questions to either GPT-4 or GPT-3.5.

By default, we train SFT on the instruction dataset with a batch size of 128 and a sequence length of 512 for 400 steps, totaling approximately 26.2 million tokens. To effectively evaluate the overall fine-tuning performance, we assess the perplexity of the fine-tuned model on an out-of-domain test set. This out-of-domain test set is constructed by randomly sampling 2,000 instances from the Alpaca (Taori et al., 2023) test set, truncated to a sequence length of 512.

Downstream Task. We focus on the performance of LORAM in various downstream tasks, including mathematical reasoning, common sense reasoning, and code generation. All our downstream task evaluations are performed on lm-evaluation-harness[5] and code-eval[6] with VLLM[7].

For mathematical reasoning, we benchmark the accuracy of baseline models using greedy decoding on MathQA (Amini et al., 2019) with a 1-shot setting and GSM8K (Grade School Math 8K) (Cobbe et al., 2021) with 8-shots, Chain of Thought (CoT) prompting and strict match MathQA is a large-scale dataset comprising 37k English multiple-choice math word problems, covering diverse math domains. It extends the AQuA-RAT dataset (Ling et al., 2017) by annotating problems with fully specified operational programs using a new representation language, building on the questions, options, rationale, and correct answers provided by AQuA-RAT. The GSM8K is a dataset of 8.5K high-quality, linguistically diverse grade school math word problems, designed to evaluate multi-step reasoning in basic arithmetic operations (+-×÷). We conduct evaluations on its 1.3K test set with *strict-match* to assess logical and mathematical reasoning in language models.

For commonsense reasoning (CSR), we report the average accuracy across six tasks—Arc Challenge & Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021)—under 1-shot and greedy decoding settings. These benchmarks comprehensively assess the model's ability to apply "commonsense" or world knowledge for reasoning, rather than relying on pattern recognition.

For code generation, we compare two pass rates, PASS@1 and PASS@10 (Kulal et al., 2019), on HumanEval (Chen et al., 2021) of each baseline in a zero-shot setting with sampling parameters of $\text{TEMPERATURE} = \{0.0, 0.2, 0.4, 0.6, 0.8\}$, and $\text{TOP}_\text{P} = 0.95$. The HumanEval dataset released by OpenAI consists of 164 handwritten Python programming problems, each with a function signature, docstring, body, and unit tests. Serving as a benchmark, HumanEval assesses models on a range of Python coding skills, from basic syntax to complex problem-solving, offering insights into their programming capabilities alongside language-focused tasks.

Sparsification & Quantization. LORAM incorporates two model compression techniques: sparsification, which generates a pruned model for low-rank matrix updates, and quantization, which forms QLORAM further to reduce the memory footprint of the pruned model. For sparsification, to validate the general effectiveness of LORAM, we benchmark its performance across various pruning strategies $\text{P}(\cdot)$. Specifically, we first establish a variant using randomly structured pruning and adapt LORAM to another three variants based on leading approaches: the structured pruning LLM-Pruner[8] (Ma et al., 2023) and the non-structured (semi-structured & unstructured) pruning SparseGPT[9] (Frantar & Alistarh, 2023a). These baselines are summarized below, with the corresponding configurations presented in Tables 4 to 6.

- **LORAM-RAND**: We adhere to the pruning settings of LORAM-STRU, modifying only by randomly removing weights instead of the original gradient-based pruning criterion.

- **LORAM-STRU**: We follow LLM-Pruner and employ a block-wise strategy for local structured pruning. Attention and MLP layers are treated as separate blocks, with non-critical coupling weights pruned based on gradient information at a uniform ratio. We retain the first four and last two layers of both blocks, focusing pruning on the intermediate layers.

---

[5] https://github.com/EleutherAI/lm-evaluation-harness (MIT License).

[6] https://github.com/abacaj/code-eval (MIT License).

[7] https://github.com/vllm-project/vllm (Apache-2.0 license).

[8] https://github.com/horseee/LLM-Pruner (Apache-2.0 license)

[9] https://github.com/IST-DASLab/sparsegpt (Apache-2.0 license)

- **LORAM-SEMI**: We utilize SparseGPT with a 4:8 semi-structured sparsity pattern to prune pre-trained weights across all model layers.

- **LORAM-UNST**: We prune individual weights uniformly across layers using a predefined pruning ratio based on an unstructured version of SparseGPT.

For quantization $\mathbb{Q}(\cdot)$, to further reduce memory usage during training, especially when dealing with models exceeding 70 billion parameters, we achieve QLORAM by combining LORAM with the LoRA-tailored quantization algorithm QLoRA (Dettmers et al., 2023). While LORAM is compatible with the quantization of other customized LoRA methods (Xu et al., 2024; Li et al., 2024; Guo et al., 2024; Frantar et al., 2023; Chai et al., 2023), this falls outside the scope of this article.

Architecture & Hyperparameters. We adopt a LLaMA architecture with RMSNorm (Zhang & Sennrich, 2019) and SwiGLU activations (Shazeer, 2020; Zhao et al., 2022). We run all experiments with BF16 format to reduce memory usage. For all configurations, we default to a learning rate of 1e-3. However, the downstream performance of models fine-tuned on OpenOrca is relatively sensitive to the learning rate. Therefore, in this evaluation, we tune the learning rates for each baseline within the range of [1e-5, 1e-3] and report their respective optimal downstream scores. Specifically, we use 1e-5 for the 7B LoRA and 13B & 70B LoRAM models, and 1e-4 for the 13B LoRA model. All experiments run on NVIDIA A100-80GB GPUs with environments of CUDA 12.2, PyTorch 2.4.0, and Transformer 4.45.1. For LLaMA-2 herds, we set low-rank matrices $\mathbf{B}$ and $\mathbf{A}$ of rank $r = 8$ for $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, and $\mathbf{W}_o$ in the attention layer, $\mathbf{W}_{up}$, $\mathbf{W}_{gate}$, and $\mathbf{W}_{down}$ in the MLP layer, and the head embedding matrix $\mathbf{W}_{lm\_head}$; for LLaMA-3 herds, we exclude the injection of the low-rank matrix of $\mathbf{W}_{lm\_head}$.

Table 4: LoRAM configures on LLaMA-2-13B. Comparison of different pruning methods in terms of parameter reduction ratio (Reduction) and HBM footprint (GB) of pruned parameters (HBM), ignoring low-rank matrix overhead.

Method	#Orig. Params	Pruning Ratio	#Pruned Params	Reduction	HBM
LoRAM-Semi	13015864320	0.50	6738415616	1.93×	12.55
LoRAM-Unst	13015864320	0.55	6037628912	2.16×	11.25
LoRAM-Rand & Stru	13015864320	0.65	6005662720	2.17×	11.19

Table 5: LoRAM configures on LLaMA-2-70B and LLaMA-3.1-70B with different pruning ratios.

Method	#Orig. Params	Pruning Ratio	#Pruned Params	Reduction	HBM
LoRAM-Rand & Stru	68976648192	0.65	28099436544	2.45×	52.34
LoRAM-Rand & Stru	68976648192	0.75	21488738304	3.21×	40.03
LoRAM-Rand & Stru	68976648192	0.85	16272924672	4.24×	30.31
LoRAM-Rand & Stru	68976648192	0.95	9662226432	7.14×	18.00
LoRAM-Rand & Stru	70553706496	0.85	17849982976	3.95×	33.25

Table 6: QLoRAM configures on LLaMA-2-70B and LLaMA-3.1-70B with , demonstrating more aggressive parameter compression.

Method	#Orig. Params	Pruning Ratio	#Pruned Params	Reduction	HBM
QLoRAM-Rand & Stru	68976648192	0.65	7024859136	9.82×	13.08
QLoRAM-Rand & Stru	68976648192	0.75	5372184576	12.84×	10.01
QLoRAM-Rand & Stru	68976648192	0.85	4068231168	16.95×	7.58
QLoRAM-Rand & Stru	68976648192	0.95	2415556608	28.56×	4.50
QLoRAM-Rand & Stru	70553706496	0.85	4462495744	15.81×	8.31

C VISUALIZATION OF DIMENSION EVOLUTION

To clearly illustrate the evolution of weight matrix dimensions across the multiple stages in the proposed scheme, we take LLM-Pruner (Ma et al., 2023) as an example in (e.g., LoRAM-STRU) in Fig. 9, visualizing the transformation from $\mathbf{W}_0 \Rightarrow \mathbf{W}_0^P$, $\mathbf{W}_\Delta \Rightarrow \mathbf{W}_\Delta^P$, and $\mathbf{W}_\Delta^{P^\star} \Rightarrow \mathbf{W}_\Delta^{R^\star}$ under LoRAM with structured pruning. For LoRAM variants employing non-structured pruning, the parameter dimensionality remains unchanged during training due to the use of a mask matrix. Therefore, these visualizations are omitted.
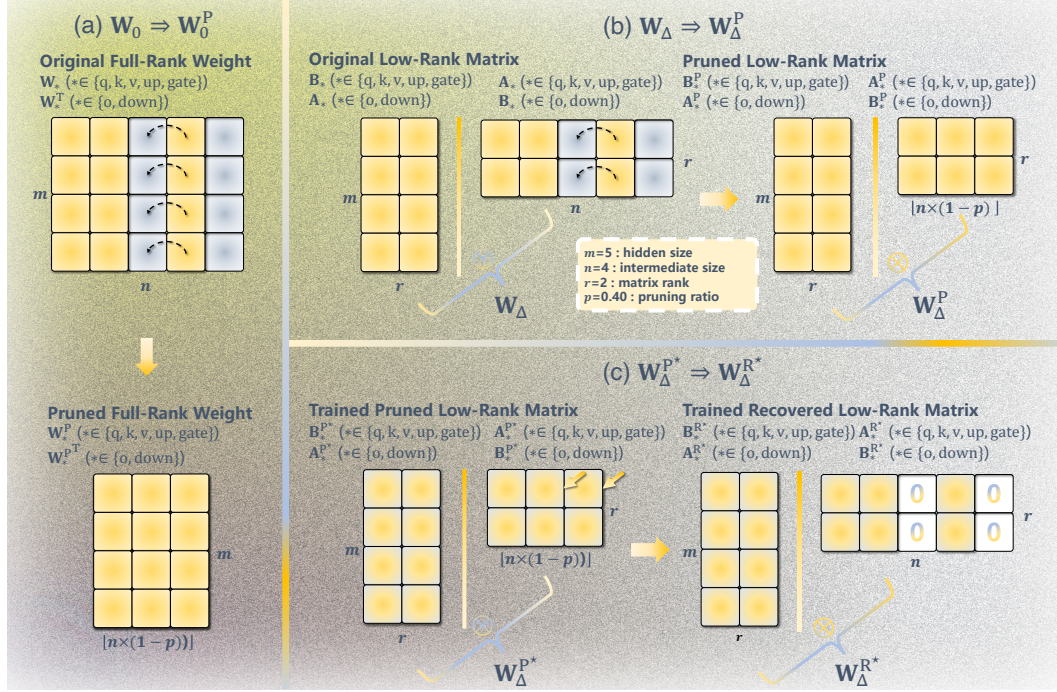

Figure 9: Dimensional evolution of the weight matrices: $\mathbf{W}_0 \Rightarrow \mathbf{W}_0^P$ (a), $\mathbf{W}_\Delta \Rightarrow \mathbf{W}_\Delta^P$ (b), and $\mathbf{W}_\Delta^{P^\star} \Rightarrow \mathbf{W}_\Delta^{R^\star}$ (c) during LoRAM-STRU training. This includes updates for $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, and $\mathbf{W}_o$ in the attention layer, as well as $\mathbf{W}_{up}$, $\mathbf{W}_{gate}$, and $\mathbf{W}_{down}$ in the MLP layer.

D VISUALIZATION OF LOW-RANK MATRICES

In this section, we utilize the L_2-norm to evaluate variations in low-rank matrices trained with different LoRAM variants. This metric facilitates the visualization of captured features and allows for an analysis of LoRAM's effectiveness. Specifically, we examine the updated low-rank matrices in the self-attention and MLP layers of LLaMA-2-13B and LLaMA-2-70B, trained with LoRAM variants on OpenHermes.

D.1 HEAD-WISE NORM OF ATTENTION

For the low-rank matrices in the attention layer, denoted as $\mathbf{W}_{\Delta^*}$ where $* \in \{q, k, v, o\}$, we compute the L_2 norms for each attention head. Let H^* represent the number of heads. The L_2 norms for each head h (where $h = 0, 1, \ldots, H^* - 1$) are defined as follows:

$$\|\mathbf{W}_{\Delta^*}^{(h)}\|_2 = \begin{cases} \|\mathbf{W}_{\Delta^*}[h, :]\|_2 & \text{if } * \in \{q, k, v\} \\ \|\mathbf{W}_{\Delta^*}[:, h]\|_2 & \text{if } * = o \end{cases}. \tag{10}$$

The results are visualized through heatmaps in Figs. 10 and 12, effectively illustrating the distribution of features captured by different attention heads.

D.2 LAYER-WISE NORM OF MLP

For the low-rank matrices in the MLP layers, denoted as $\mathbf{W}_{\Delta^*}$ where $\Delta^* \in \{up, gate, down\}$, we denote the number of layers as L. The average L_2 norm for a specific layer l (where $l = 0, 1, \ldots, L - 1$) is computed as follows, excluding elements equal to zero using a mask, ensuring that only active parameters contribute to the average:

$$\|\mathbf{W}_{\Delta^*}^{(l)}\|_2 = \begin{cases} \frac{1}{m} \sum_{i=0}^{m-1} \left\|\mathbf{W}_{\Delta^*}^{(l)}[i, :]\right\|_2 \cdot \mathbb{I}(\mathbf{W}_{\Delta^*}^{(l)}[i, :] \neq 0) & \text{if } \Delta^* \in \{up, gate\} \\ \frac{1}{n} \sum_{j=0}^{n-1} \left\|\mathbf{W}_{\Delta^*}^{(l)}[:, j]\right\|_2 \cdot \mathbb{I}(\mathbf{W}_{\Delta^*}^{(l)}[:, j] \neq 0) & \text{if } \Delta^* = down \end{cases}. \tag{11}$$

Here, $\mathbb{I}(\cdot)$ denotes the indicator function, which returns 1 only when the corresponding element is non-zero, effectively excluding zero elements from the average calculation. The average norms for the MLP layers are visualized in Figs. 11 and 13, clearly depicting the trends in updating amplitudes across the various projections.

D.3 ATTENTION UPDATE PATTERNS

Layer Update Patterns in LoRAM and LoRA. Figs. 10 and 12 reveal that both LoRA and LoRAM display similar layer update behaviors. In any low-rank matrix $\mathbf{W}_{\Delta^*}$ where $* \in \{q, k, v, o\}$, deeper colors predominantly concentrate in either shallow or deep layers, while middle layers receive relatively few updates. This suggests that training primarily focuses on optimizing the shallow layers to capture semantic information, with deeper layers refining this knowledge, rendering middle layers somewhat redundant.

More Uniform Projection Updates in LoRAM. Figs. 10 and 12 further indicates that updates in the LoRA-trained low-rank matrices, particularly for $\mathbf{W}_{\Delta^v}$, are relatively uniform, exhibiting substantial deep colors across multiple heads. In contrast, other matrices emphasize specific rows and heads. For instance, in the 70B model's $\mathbf{W}_{\Delta^k}$, only the heads in the uppermost layers experience significant updates, while lower layers show minimal changes. This suggests that the unpruned model retains rich knowledge, requiring only minor adjustments to a few heads in certain layers for task adaptation. Conversely, LoRAM demonstrates a more uniform distribution of deep colors across each low-rank matrix, indicating that the pruned model must effectively utilize every limited neuron to capture knowledge, thereby enhancing downstream performance.
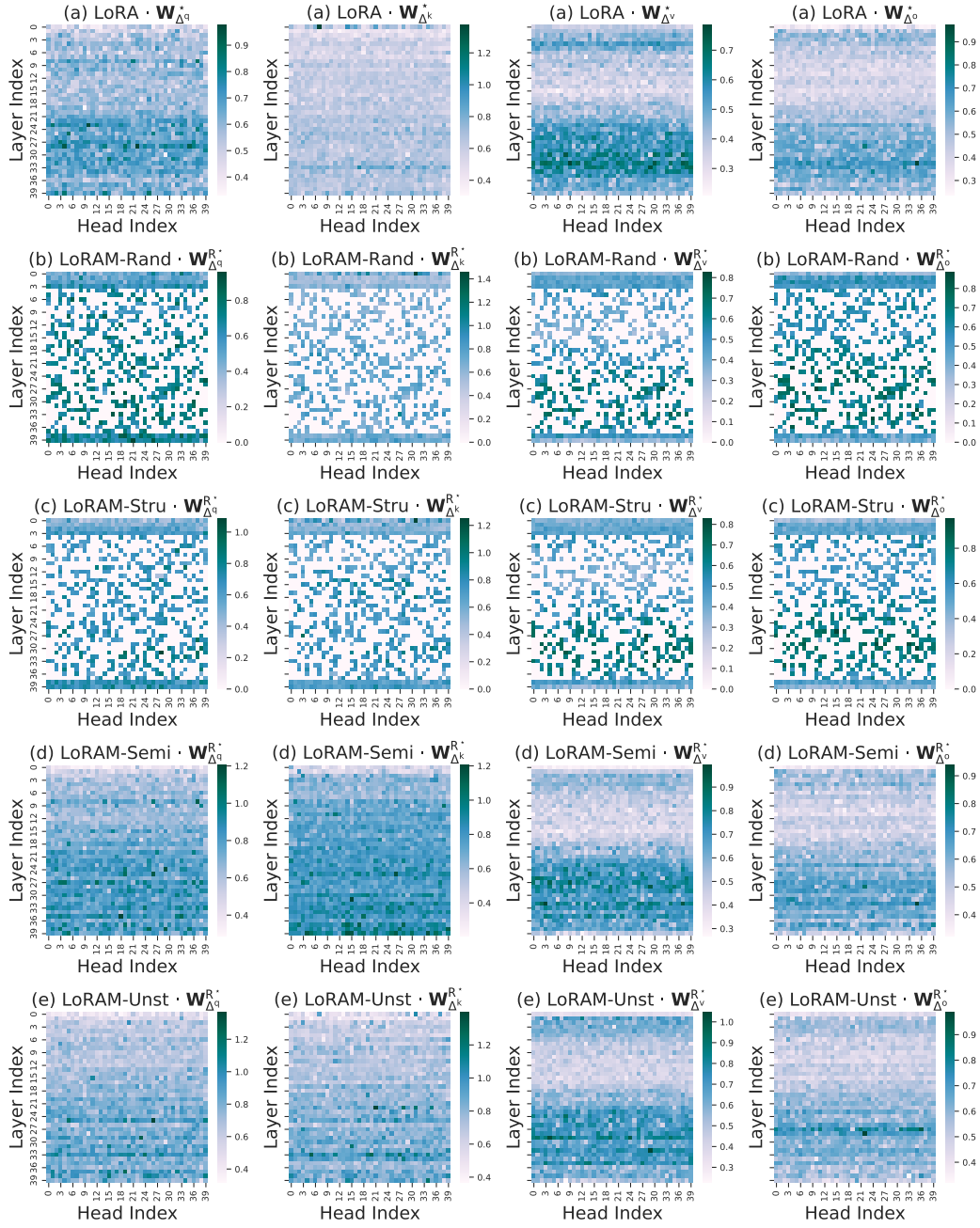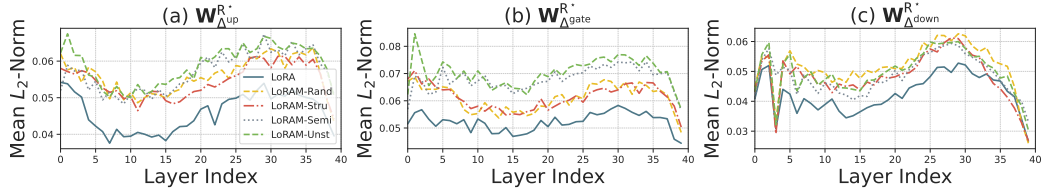
Figure 10: Visualization of low-rank matrices in the attention layers of LLaMA-2-13B.

Figure 11: Average L_2 norms of low-rank matrices in the MLP layers of LLaMA-2-70B.
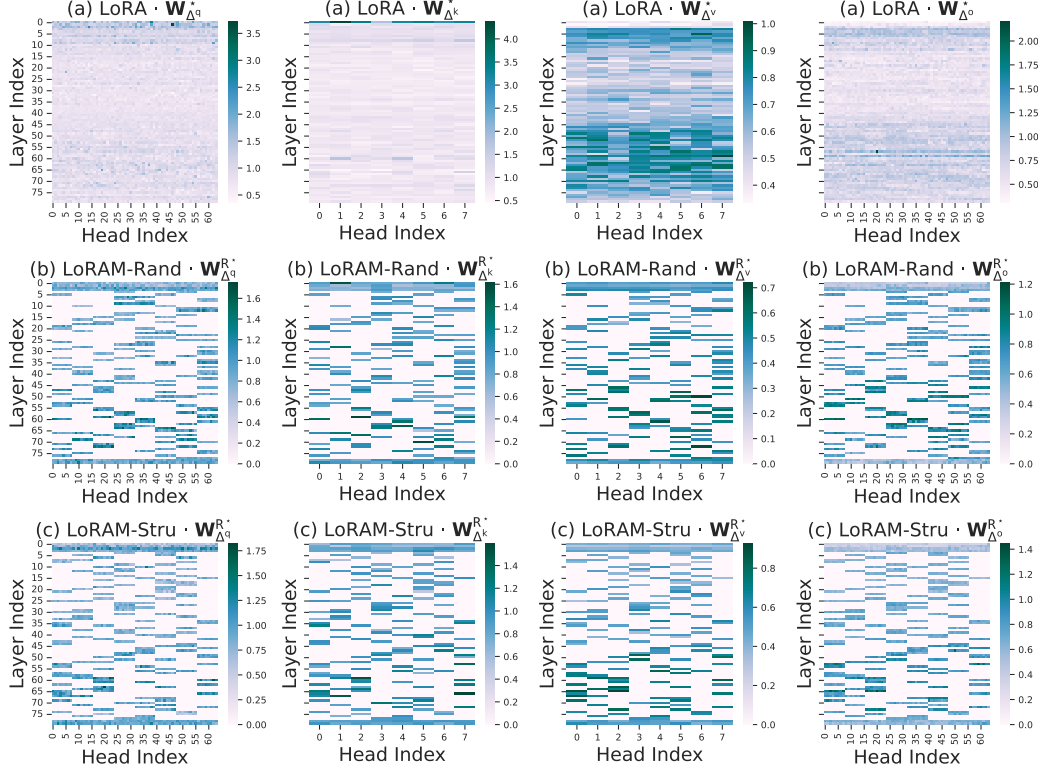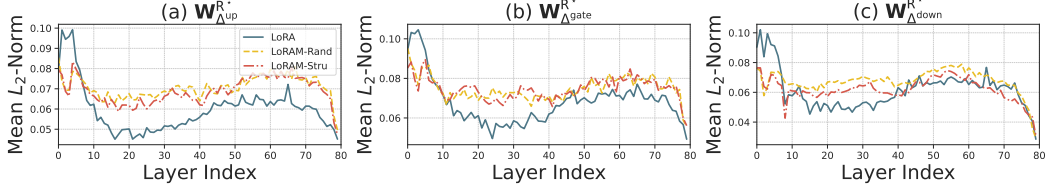
23

Figure 12: Visualization of low-rank matrices in the attention layers of LLaMA-2-70B.

Figure 13: Average L_2 norms of low-rank matrices in the MLP layers of LLaMA-2-70B.

D.4 MLP UPDATE PATTERNS

LoRAM Exhibits Greater Update Amplitude than LoRA. For both the 13B and 70B models, LoRAM consistently exhibits a greater update amplitude across each layer compared to LoRA, as shown in Figs. 11 and 13. This increased amplitude indicates that LoRAM is more effective in adjusting the weights in all layers, thus enhancing the adaptability and overall performance.

Distinct Update Trends in Layer Amplitudes. The amplitude changes reveal a distinct pattern in Figs. 11 and 13: first decreasing, then increasing, and finally decreasing again. Shallow layers (0-3) and deeper layers (25-35 for the 13B model and 50–75 for the 70B model) undergo intensive updates. This behavior indicates that model prioritizes foundational feature extraction in shallow layers and the refinement of complex representations in deeper layers. Such a strategic update distribution optimizes the learning process, ensuring effective capture of basic and advanced features.

D.5 ANALYSIS OF UNCHANGED WEIGHTS

Here, we try to analyze the unchanged weights to support the motivation of LoRAM.

Fine-Grained Visualizations. As the above visualization, we conducted detailed visualizations comparing the updated magnitudes of pruned and unpruned weights across layers. The results demonstrate that unpruned weights in both attention and MLP layers exhibit consistently smaller updates during fine-tuning as shown in Fig. 12, indicating their critical role in preserving the model's capacity for inference.

Theoretical Perspective. The phenomenon can be explained by the gradient-based importance of these weights, which prioritize parameters with minimal updates but high sensitivity during recovery. These weights stabilize inference outputs, making them indispensable despite their limited fine-tuning updates.

Quantitative Evidence Our analysis reveals a strong correlation between weight update magnitudes and downstream performance. Pruning weights with smaller updates significantly degrades performance, highlighting their importance for inference and validating our intuition.

Impact on Large Models The selective pruning strategy shows notable benefits in larger models such as LLaMA-2-70B, where it outperforms random pruning by a substantial margin. Retaining critical parameters ensures effective task adaptation and generalization across diverse domains.

E PERFORMANCE OF SUB-TASKS IN CSR

We report the performance of six sub-tasks in CSR, with Figs. 14 and 15 showcasing the results for LORAM-trained LLaMA-2-13B and LLaMA-2-70B, respectively. Our findings indicate that various LORAM variants outperform core competitive benchmarks: for the 13B model, LORAM surpasses both the untrained 13B and the LoRA-trained 7B, while for the 70B model, it exceeds the untrained 70B and the LoRA-trained 13B. This demonstrates that LORAM consistently achieves performance gains across models of different scales while effectively reducing memory usage. Furthermore, selective weight contributions in the 70B model significantly enhance performance, as evidenced by LORAM-STRU's marked improvement, particularly in the challenging Arc Challenge multi-choice question-answering task. This suggests that LORAM-STRU effectively identifies and leverages weight differences, focusing on the most trainable weights compared to LORAM-RAND.
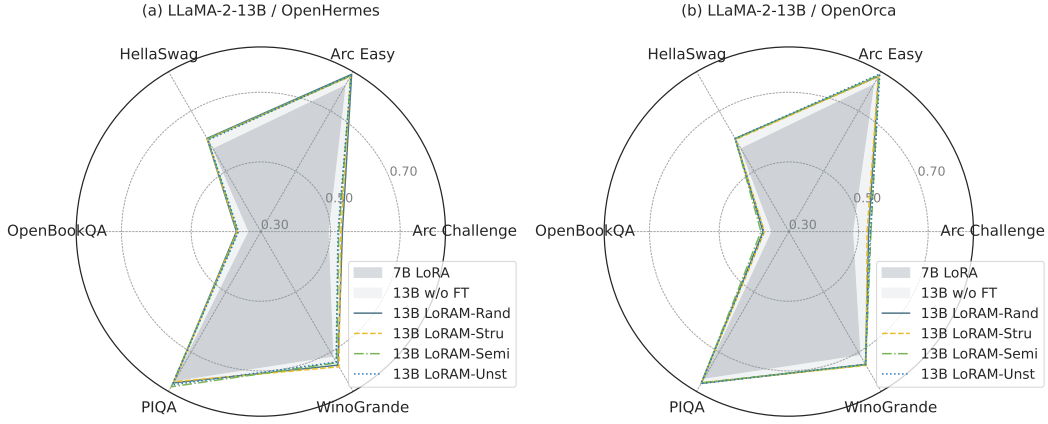

Figure 14: Performance of six CSR sub-tasks on the trained LLaMA-2-13B using LORAM.
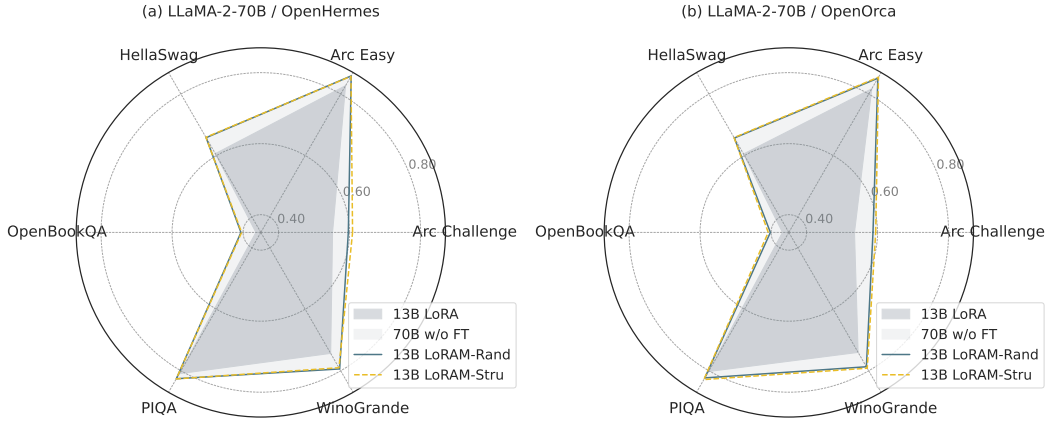

Figure 15: Performance of six CSR sub-tasks on the trained LLaMA-2-70B using LORAM.

F ALGORITHM OF LoRAM

Here, we present the complete algorithm of LoRAM in Algorithm 1.

Algorithm 1 LoRAM (Memory-Efficient LoRA Training)

Require: original full-rank pre-trained weight $\mathbf{W}_0$, alignment corpus $\mathcal{D}_{\mathtt{A}}$, and flags $\mathcal{F}^{\mathtt{P}}, \mathcal{F}^{\mathtt{A}}, \mathcal{F}^{\mathtt{Q}}, \mathcal{F}^{\mathtt{R}}$.
1: **Offline $\mathbf{W}_0^*$ Process Stage:**
2: **if** $\mathcal{F}^{\mathtt{P}}$ **then**
3:      $\mathbf{W}_0^{\mathtt{P}} = \mathtt{P}(\mathbf{W}_0) = \mathbf{W}_0 \circ \mathbf{M}^{\mathtt{P}}$ ▷ Pruned Full-Rank Weight Generation.
4:      **if** $\mathcal{F}^{\mathtt{A}}$ **then**
5:          $\mathbf{W}_0^{\mathtt{P,A}} \leftarrow \arg\min \mathcal{L}_{\mathtt{A}}(\mathcal{D}_{\mathtt{A}}; \mathbf{W}_0^{\mathtt{P}})$ ▷ Pruned Full-Rank Weight Alignment.
6:          **if** $\mathcal{F}^{\mathtt{Q}}$ **then**
7:              $\mathbf{W}_0^{\mathtt{P,A,Q}} = \mathtt{Q}(\mathbf{W}_0^{\mathtt{P,A}})$ ▷ Pruned Full-Rank Weight Quantization.
8:          **end if**
9:      **else if** $\mathcal{F}^{\mathtt{Q}}$ **then**
10:          $\mathbf{W}_0^{\mathtt{P,Q}} = \mathtt{Q}(\mathbf{W}_0^{\mathtt{P}})$
11:      **end if**
12: **else if** $\mathcal{F}^{\mathtt{Q}}$ **then**
13:      $\mathbf{W}_0^{\mathtt{Q}} = \mathtt{Q}(\mathbf{W}_0)$ ▷ Standard Quantization for LoRA
14: **end if**
15: Record the processing result of $\mathbf{W}_0$ as $\mathbf{W}_0^*$, $* \in \{\mathtt{NULL}, \mathtt{P}, \mathtt{Q}, (\mathtt{P}, \mathtt{Q}), (\mathtt{P}, \mathtt{A}), (\mathtt{P}, \mathtt{A}, \mathtt{Q})\}$.
16:
17: **Online $\mathbf{W}_\triangle^*$ Training Stage:**
18: **if** $\mathcal{F}^{\mathtt{P}}$ **then**   ▷ Pruned Low-Rank Matrix Generation.
19:      $\mathbf{W}_\triangle^{\mathtt{P}} = \mathbf{B}^{\mathtt{P}}\mathbf{A}^{\mathtt{P}} = \mathtt{P}(\mathbf{W}_\triangle) = \mathbf{W}_\triangle \circ \mathbf{M}^{\mathtt{P}} = \mathbf{B}\mathbf{A} \circ \mathbf{M}^{\mathtt{P}}$
20:      **while** TRAINING **do**   ▷ Pruned Low-Rank Matrix Training.
21:          Update low-rank matrix via objective $\mathcal{L}_{\mathrm{SFT}}$ with the forward pass $\mathbf{h} = \mathbf{x}\mathbf{W}_0^* + \mathbf{x}\mathbf{W}_\triangle^{\mathtt{P}}$.
22:          Return trained low-rank matrix $\mathbf{W}_\triangle^{\mathtt{P}^\star} = \mathbf{B}^{\mathtt{P}^\star}\mathbf{A}^{\mathtt{P}^\star}$.
23:      **end while**
24:      **if** $\mathcal{F}^{\mathtt{R}}$ **then**   ▷ Recovered Low-Rank Matrix Generation.
25:          $\mathbf{W}_\triangle^{\mathtt{R}^\star} = \mathbf{B}^{\mathtt{R}^\star}\mathbf{A}^{\mathtt{R}^\star} = \mathtt{R}(\mathbf{W}_\triangle^{\mathtt{P}^\star}) = \mathbf{W}_\triangle^{\mathtt{P}^\star} \circ (1 - \mathbf{M}^{\mathtt{P}})$ ▷ Structured LoRAM
26:      **else**
27:          $\mathbf{W}_\triangle^{\mathtt{R}^\star} = \mathbf{B}^{\mathtt{R}^\star}\mathbf{A}^{\mathtt{R}^\star} = \mathbf{B}^{\mathtt{P}^\star}\mathbf{A}^{\mathtt{P}^\star}$ ▷ Non-structured LoRAM
28:      **end if**
29: **else**
30:      **while** TRAINING **do**   ▷ Standard LoRA Training.
31:          Update low-rank matrix via objective $\mathcal{L}_{\mathrm{SFT}}$ with the forward pass $\mathbf{h} = \mathbf{x}\mathbf{W}_0^* + \mathbf{x}\mathbf{W}_\triangle$.
32:          Return trained low-rank matrix $\mathbf{W}_\triangle^\star = \mathbf{B}^\star\mathbf{A}^\star$.
33:      **end while**
34: **end if**
35: Record the trained low-rank matrix as $\mathbf{W}_\triangle^*$, $* \in \{\mathtt{R}^\star, \star\}$.
36:
37: **Online $\mathbf{W}_0, \mathbf{W}_\triangle^*$ Inference Stage:**
38: **while** INFERENCE with $*$ is $\mathtt{R}^\star$ **do**   ▷ Recovered Low-Rank Matrix Inference.
39:      Perform inference with the forward pass $\mathbf{h} = \mathbf{x}(\mathbf{W}_0 + \mathbf{W}_\triangle^{\mathtt{R}^\star}) = \mathbf{x}(\mathbf{W}_0 + \mathbf{B}^{\mathtt{R}^\star}\mathbf{A}^{\mathtt{R}^\star})$.
40: **end while**
41: **while** INFERENCE with $*$ is $\star$ **do**   ▷ Standard LoRA Inference.
42:      Perform Inference with the forward pass $\mathbf{h} = \mathbf{x}(\mathbf{W}_0 + \mathbf{W}_\triangle^\star) = \mathbf{x}(\mathbf{W}_0 + \mathbf{B}^\star\mathbf{A}^\star)$.
43: **end while**

G TUNING OF LEARNING RATE

We provide additional details on the learning rate tuning process for full LoRA applied to LLaMA-2-7B and LLaMA-2-13B models, trained on the OpenHermes dataset. These experiments in Fig. 16 demonstrate that a learning rate of 1e-3 consistently achieves the best perplexity across both in-domain and out-of-domain datasets, further validating the reliability of our comparison.
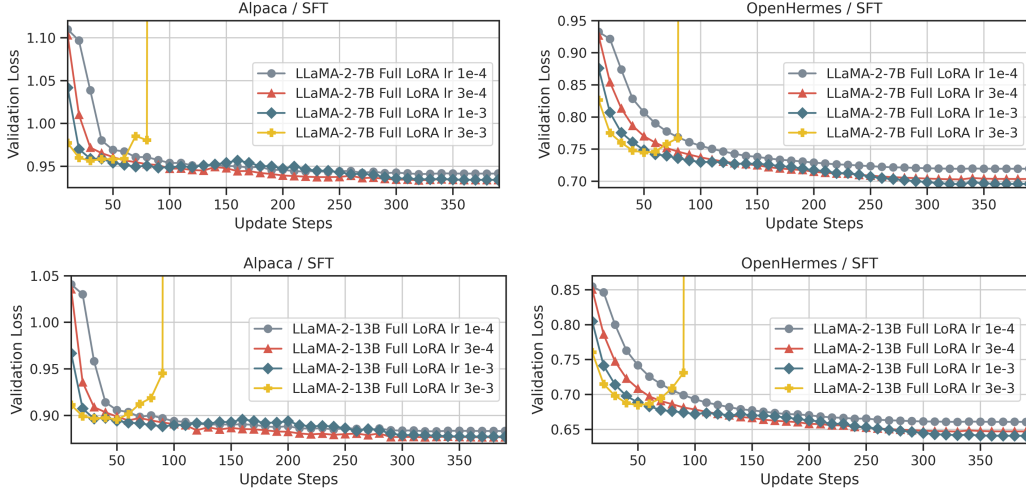

Figure 16: Learning rate tuning for LLaMA-2-7B and LLaMA-2-13B on OpenHermes using LoRA.

H PERFORMANCE OF DOMAIN-SPECIFIC TASK

To assess the effectiveness of LoRAM in domain-specific tasks, we conducted experiments on GSM8K (using the training set for tuning and the test set for evaluation), a mathematical reasoning benchmark known for its sensitivity to sparsification. Specifically, we trained LLaMA-3.1-70B using QLoRAM under various configurations.

The results, summarized in Table 7, highlight that LoRAM achieves excellent performance in this domain-specific setting. Notably, LoRAM-based models maintain high accuracy with substantial parameter reduction ratios, showcasing their robustness and efficiency in domain-specific tasks. These findings emphasize LoRAM's broad applicability beyond general-purpose instruction fine-tuning.

Table 7: Evaluation of LoRAM on the GSM8K dataset for domain-specific fine-tuning. Results show accuracy (%) and parameter reduction ratios for different configurations.

LLaMA-3.1	GSM8K	Parameter Reduction Ratio
8B w/o Fine-Tuning	55.27	8.79×
8B LoRA (OpenHermes 400)	55.80	8.79×
70B w/o Fine-Tuning	75.28	1.00×
70B QLoRAM-Stru 400 (OpenHermes 400)	80.36	**15.81×**
70B QLoRAM-Stru 400 (GSM8K 100)	77.18	**15.81×**
70B QLoRAM-Stru 400 (GSM8K 200)	79.15	**15.81×**
70B LoRA (OpenHermes 400)	**80.74**	1.00×

I ANALYSIS OF LORAM COST

Identifying the costs of LoRAM is indeed important, which is why we report both the number of training tokens used during the alignment phase and the parameter reduction ratios in the low-rank training phase. Below, we clarify the two stages of LoRAM:

Offline Knowledge Alignment Phase. The offline phase is task-agnostic and can be conducted by the model publisher prior to deployment, making its cost negligible for end users. To quantify the offline cost, we measured the number of training tokens (as in Xia et al. (2024)) rather than end-to-end latency, which can vary based on hardware configurations. As shown in Figure 5, LoRAM achieves significant performance gains using only 13 million tokens, demonstrating the efficiency of the alignment phase.

Online Low-Rank Matrix Training Phase. For the online phase, the memory and latency costs are primarily determined by the size of the base model parameters, which dominate resource consumption during training. To avoid redundancy in reporting, we focused on parameter reduction ratios instead of absolute time or memory usage.

Comparative Metrics for Online Training. Here, we provide additional metrics, including memory and latency comparisons for the online training phase. We conducted experiments using a workload of 1024 samples (batch size 128, micro-batch size 4, sequence length 512) randomly selected from OpenHermes. The results in Table 8 demonstrate that LoRAM with a structured pruning ratio of $2.17\times$ (13B $\rightarrow$ 6B) achieves comparable peak memory, latency, and throughput to 7B LoRA, with only minor trade-offs. These differences arise due to the larger layer count in 13B LoRAM, introducing more non-GEMM operations, slightly affecting latency and throughput.

These results underscore the advantages of LoRAM's design in achieving substantial resource efficiency without significant trade-offs in memory or latency.

Table 8: Comparison of peak memory (MiB), latency (s), and throughput (samples/s) during the online training phase for LoRAM and LoRA models. Results are based on a workload of 1024 samples (batch size 128, micro-batch size 4, sequence length 512).

LLaMA-2	#Model Params	Reduction Ratio	Memory	Latency	Throughput
7B LoRA	6.73B	1.93×	30,517	**134.27**	**7.626**
13B LoRA	13.02B	1.00×	51,661	206.07	4.969
13B LoRAM-Stru	**6.01B**	**2.17×**	**29,799**	147.86	6.925

J ANALYSIS OF CHANGES IN PERFORMANCE TRENDS

We analyze performance at two stages: after fine-tuning but before recovery, and after both fine-tuning and recovery.

After Fine-Tuning but Before Recovery. At this stage, the results of LoRAM align with prior work (e.g., SparseGPT, Wanda, and LLM-Pruner). Unstructured and semi-structured pruning consistently outperform structured pruning (see Fig. 6, solid lines). This trend holds true across both aligned and unaligned settings, with the performance order as follows: LoRAM-SEMI < LoRAM-UNST < LoRAM-STRU < LoRAM-RAND The slight advantage of LoRAM-SEMI over LoRAM-UNST can be attributed to its smaller pruning ratio, which retains more parameters and mitigates performance degradation.

After Fine-Tuning and Recovery. Post-recovery results show that structured pruning outperforms unstructured pruning. This can be explained by two factors:

- **Preserved Structure for Recovery:** Structured pruning maintains the organization of the pruned weights into coherent structures (e.g., rows and columns in MLP layers, attention heads in attention layers), ensuring that activations after recovery are aligned with those of the original model. This alignment improves the recovery process.

- **Pruned Weight Quality:** The quality of pruned weights influences the recovery effectiveness. Structured pruning tends to remove less critical weights, leaving more recoverable parameters. In contrast, unstructured pruning can remove weights that are more difficult to recover, which negatively impacts performance post-recovery.

These results highlight the interplay between pruning and recovery, suggesting that structured pruning, despite initial performance disadvantages, facilitates more effective recovery.