

# LoRA Decompose: Serving Fine-Tuned Models into LoRA-Like

Yibo Han<sup>a,\*</sup>, Tangzhi Xu<sup>b</sup>, Zenan Li<sup>b</sup>, Yijia Zhang<sup>a</sup>, Youshan Miao<sup>c</sup>, Yuan Yao<sup>b</sup> and Ningyi Xu<sup>a,\*\*</sup>

<sup>a</sup>Shanghai Jiao Tong University

<sup>b</sup>Nanjing University

<sup>c</sup>Microsoft Research

**Abstract.** Large language models (LLMs) achieve remarkable performance across diverse tasks but face increasing GPU-memory demands due to the growing variety and complexity of downstream tasks. Efficient inference has thus become essential, especially for resource-limited settings.

In this paper, we propose LoRA Decompose, a novel compression approach based on a key insight: instruction-fine-tuned models share a common pretrained-like base component and differ primarily through low-rank, LoRA-like delta components. Leveraging this observation, we reformulate the inference problem as a constrained optimization task that jointly identifies a shared low-rank structure across multiple models, significantly reducing their memory footprints.

We solve this optimization efficiently using a custom-designed block coordinate descent algorithm, converging quickly within a few iterations. Empirical experiments with Llama-2 7B and 13B models demonstrate that our method achieves a remarkable >32x GPU memory reduction while preserving task accuracy, allowing substantial efficiency gains for practical deployment.

## 1 Introduction

Large language models (LLMs) have shown surprising performance in various tasks, including language comprehension [9], code generation [5], and math problem solving [16]. To achieve such abilities, it is a common practice to employ a "pretrain then fine-tune" paradigm. With the process of fine-tuning, task-specific models demonstrate substantially superior performance compared to general pretrained models. For instance, in the human-eval benchmark (a code generation task), code-llama 7B model can even outperform llama-2 70B model by a large margin, achieving a top-1 accuracy of 33.5% compared to 30.5%. Notably, Code-Llama 7B is an order of magnitude smaller than LLaMA-2 70B, underscoring the efficiency and effectiveness of task-specific fine-tuning. [1]

To adapt large language models to specific downstream tasks, two main approaches are commonly used: full-parameter fine-tuning and Low-Rank Adaptation (LoRA). Full-parameter finetuning offers maximum performance by updating all model weights, making it ideal for tasks with significant distribution shifts from pretraining. However, shown in Figure 1(a), it demands substantial computational and storage resources, posing challenges for deployment in resource-constrained environments. LoRA[11, 15, 6, 12], a parameter-efficient

fine-tuning (PEFT) method, addresses these limitations by introducing trainable low-rank updates while keeping most pretrained weights fixed, significantly reducing resource consumption with minimal performance trade-off. In deployment, LoRA enables efficient multi-model inference with low memory overhead, as shown in Figure 1(b). Nonetheless, in tasks requiring extreme precision or major structural modifications, LoRA may underperform compared to full-parameter finetuning.

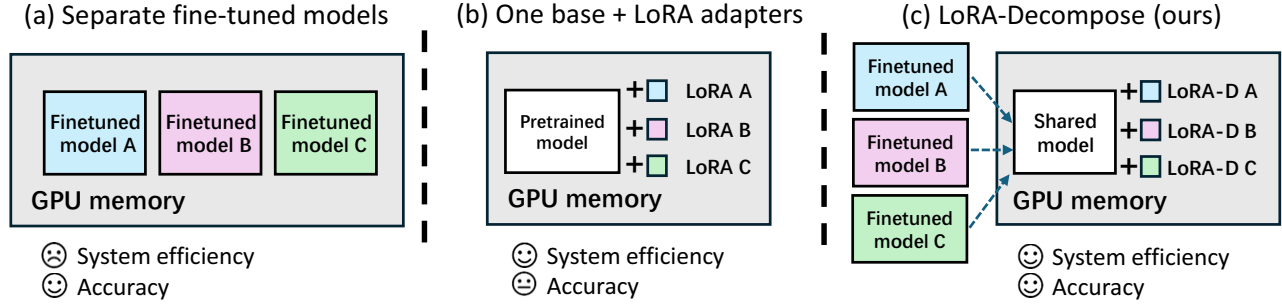
**Why collaboratively multi-model inference is important?** The vision of deploying a large number of fine-tuned LLMs, each specialized for different tasks or user preferences, is increasingly compelling. Recent advances such as parameter-efficient fine-tuning and quantization techniques have made it feasible to train and store many such models even on consumer-grade GPUs [6]. However, realizing this vision in practice remains challenging. Once fine-tuned, hosting these models independently on dedicated GPUs becomes prohibitively expensive in terms of both hardware and energy. This challenge is especially pronounced in LLM agent systems [2, 10, 23], which benefit from invoking multiple specialized models for distinct subtasks. To ensure low-latency responses, these models must be co-located and served collaboratively, often through concurrent or parallel execution [21]. Despite its importance, this problem setting remains underexplored, with only a few recent efforts addressing the need for efficient collaborative multi-model inference [14, 19].

Current optimization techniques show limited effectiveness in reducing memory usage, as most studies concentrate on single-model quantization. Since FP16 remains the standard inference format and the most aggressive form of quantization compresses models to 1-bit representations [14], these methods achieve at most a 16x reduction in size. Moreover, such approaches largely neglect opportunities for collaborative optimization across multiple models. To address this limitation, we introduce a novel framework that exploits the shared knowledge and low-rank structure inherent in collections of fine-tuned models derived from a common pretrained model.

A straightforward method is to directly truncate the rank of delta weight between pretrained model and fine-tuned model. But as shown in our experiments that this straightforward method badly hurts the performance of the model. To this end, we propose our method LoRA Decompose. Our intuition is to collaboratively compress the fine-tuned models and find an optimum shared base weight and delta counterparts for each fine-tuned model to be low-rank. Our observation shows that the delta counterparts do show the property of low-rank.

\* Email: harryhan618@sjtu.edu.cn

\*\* Corresponding author: xuningyi@sjtu.edu.cn



**Figure 1:** Methods comparison. Our proposed method LoRA Decompose has good system efficiency and accuracy.

Shown in Figure 1(c), our approach introduces a collaborative compression strategy that decomposes each fine-tuned model into a shared base component and individual delta components, enabling efficient storage and inference. Through extensive experiments on diverse model collections, we demonstrate the superior performance of our method compared to state-of-the-art single-model compression techniques. The key contributions of this work can be summarized as follows:

1. We propose a novel collaborative compression framework for fine-tuned model collections, exploiting their shared knowledge and low-rank structure.
2. We introduce a layer-wise decomposition approach that effectively isolates common and unique information across models, enabling efficient compression.
3. We conduct extensive experiments on various model collections, demonstrating the superiority of our method over existing single-model compression techniques.
4. We provide a comprehensive analysis of the factors influencing collaborative compression performance, offering valuable insights for future research.

In conclusion, Section 2 provides the motivation and Problem Statement to justify this new usage scenario and its main challenge. Section 3 extends our collaborative compression framework. Section 4 shows our experiments on compressing multiple Llama-2 based fine-tuned models on different downstream tasks. Section 5 and Section 6 are related work and discussions respectively.

## 2 Motivation and Problem Statement

**Motivation** Current research in large language models (LLMs) has not fully addressed the potential of multi-model collaborative inference. However, this scenario is not hypothetical. The "pretrain then fine-tune" paradigm prompts LLM users to fine-tune their own task-specific models based on open source pre-trained models. This practice spurs many applications like LLM-based agents that involve multi-model inference. For example, HuggingGPT[21] employs a general LLM as a task planner and subsequently uses expert models from HuggingFace to execute individual steps. Since the number of expert models is uncertain and their types in use is indefinite as well, it is impossible to allocate ad-hoc GPU for each model. These applications demand co-location of multiple models on few proximate GPUs or even just one GPU for efficient service deployment.

**Challenge** In this context, the primary challenge is accommodating multiple models onto constrained GPU resources with limited memory, especially when these applications are more likely to be deployed on the edge side. Even Nvidia's GeForce RTX 4090, the most high-end consumer-grade GPU, with 24GB of memory, can only

hold up to two typical fine-tuned models (7B to 13B parameters) when using fp16 format for inference. This calculation only accounts for parameter memory, excluding other essential memory requirements such as activations, KV cache, and other components. Consequently, to enable those future applications, effective optimization strategies are necessary to mitigate this storage bottleneck.

In this paper, we aim to explore and address this challenge, paving the way for broader applications of multi-model collaboration in real-world scenarios. We present LoRA Decompose, which collaboratively compresses multiple models into a shared base model and respective low-rank delta components. Our method effectively enhances the ability to store and deploy a number of models. Compared to related works such as quantization methods [13, 8, 14], our method is more memory efficient and accurate.

We can further deploy these models for parallel inference thanks to SLoRA and Punica[4]. Since our method transforms multiple fine-tuned models into LoRA-like models with a shared base model, it is straightforward to exploit their designed system by batching requests for different models using the multi-LoRA kernel. This kernel packs all the requests for different LoRA models in one batch and conducts the computation in one kernel launch, so that different models would not need to switch between GPU on and off chip at an expensive cost. Since our method is designed to transform models into LoRA-like decompositions, it is inference-friendly.

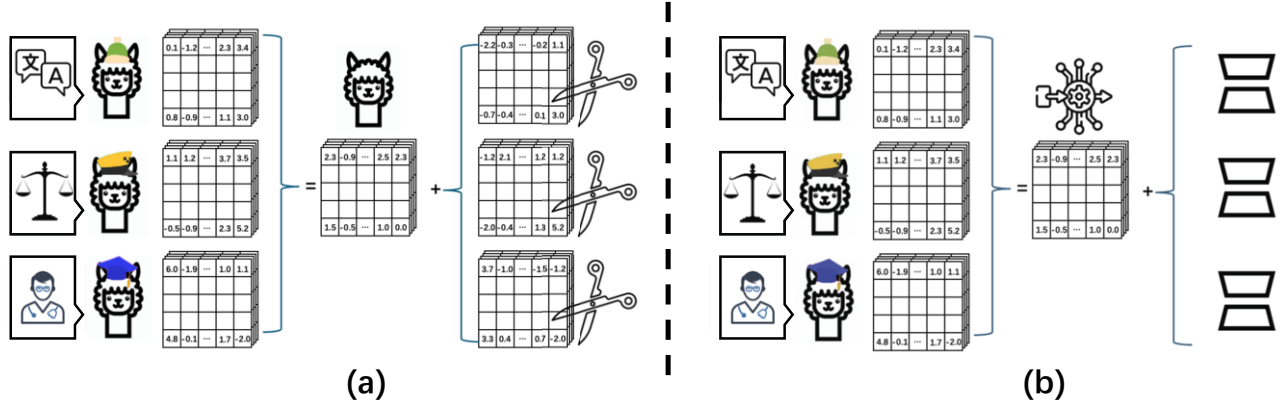
To this end, our research will not only enable multi-model collaborative inference on consumer-grade GPUs but also unlock new possibilities for deploying complex AI applications in resource-constrained environments, ultimately democratizing access to advanced AI capabilities.

## 3 Method

In this section, we present our methodology for compressing multiple models. We begin by formulating the compression task as an optimization problem, establishing the foundation for analysis. As a starting point, we introduce a direct solution that serves as a baseline, though it yields sub-optimal performance. To overcome this limitation, we propose LoRA Decompose, which employs the block coordinate descent (BCD) algorithm to efficiently optimize the objective. Importantly, we show that the direct method is a special case of our generalized framework. An overview of both methods is provided in Figure 2. Finally, we provide the theoretical analysis to guarantee the convergence of our approach.

### 3.1 Formulation

To start with, we aim to discover the latent low-rank structure that may present among a series of fine-tuned models. Specifically, we consider a set of models  $\mathcal{M}_1, \dots, \mathcal{M}_K$ , each derived from the same



**Figure 2:** (a). Direct Delta Compression baseline, in which delta matrices are directly pruned to low-rank form. (b). Our proposed method, LoRA Decompose, which optimizes a shared base matrix across models. .

pretrained model  $\mathcal{P}$  through fine-tuning. For example, both "meta-llama/Llama-2-7b-chat-hf" model and "lmsys/vicuna-7b-v1.5" model are all fine-tuned versions from "meta-llama/Llama-2-7b-hf" model.

Rather than compressing each fine-tuned model individually, we propose a collaborative approach. Our intuition is that the fine-tuning process introduces relatively minor changes compared to the vast amount of information already encoded in the pretrained model. Consequently, we hypothesize that there exists a common base model, from which the deviation of each fine-tuned model can be represented within a low-dimensional space. This low-rank structure enables an effective framework to compress the entire collection of fine-tuned models simultaneously.

In other words, we decompose each fine-tuned model into a shared base component and a unique delta component. Since these models share the same architecture, we can decompose and compress each model in a layer-by-layer manner.

Formally, consider a single parameter layer within a transformer model, such as a query projection layer (Q) or a feedforward network (FFN) linear layer. Let  $\{W_i\}_{i=1}^K$  represent the original weight matrices of this layer across these  $K$  fine-tuned models. Our goal is to find a common base matrix  $B$  and a set of low-rank delta matrices  $\{L_i\}_{i=1}^K$  such that each weight matrix  $W_i$  can be approximated as  $W_i \approx B + L_i$ .

We use the Frobenius norm to quantify the approximation error between the original matrices  $W_i$  and the approximated matrices  $\hat{W}_i = B + L_i$ . Meanwhile, we adopt the nuclear norm as a convex relaxation to impose a low-rank penalty.

$$\min_{B, L_0, \dots, L_K} \sum_{i=1}^K \left( \frac{1}{2} \|W_i - B - L_i\|_F^2 + \rho_i \|L_i\|_* \right) \quad (1)$$

### 3.2 Direct Delta Compression Baseline

Before looking into our proposed solution, we first establish a baseline approach. Following our intuition, it is easy to assume that the direct deviation from the pretrained model to each fine-tuned model can be effectively represented within a low-dimensional space. Note that this is the very assumption that is adopted in Bitdelta [14], where one-bit quantization is applied to this direct deviation for model compression.

Consequently, the base matrix  $B$  is set to be the pretrained model matrix  $P$ . We then seek to find the solutions for  $L_0, \dots, L_K$  in problem 1. The most straightforward practice in this baseline involves performing a low-rank approximation for each delta matrix, where  $D_i = W_i - P$ . In detail, each  $D_i$  undergoes singular value decom-

position (SVD), resulting in  $D_i = U\Sigma V^T$ . Here,  $\Sigma$  is a rectangular diagonal matrix which contains the singular value of  $D_i$ , represented as  $D_i = \sum_{j=1}^d \sigma_{ij} u_{ij} v_{ij}^T$ , where the singular values  $\sigma_{i1}, \dots, \sigma_{id}$  are sorted in descending order. By manually setting a target rank  $r$ , we obtain the low-rank approximation  $\hat{D}_i = \sum_{j=1}^r \sigma_{ij} u_{ij} v_{ij}^T$ . Finally, the compressed weight matrix is calculated as  $\hat{W}_i = P + \hat{D}_i$ .

In our experiments 1 and 2. We show that this brutal cutoff of delta models is harmful to their model accuracy.

### 3.3 The proposed method

We solve this problem through the block coordinate descent (BCD) algorithm [18], which alternatively minimizes the base matrix  $B$  and the lora-rank delta matrices  $L_i$  in each iteration.

---

#### Algorithm 1 LoRA Decompose

---

- 1: **Input:** A set of fine-tuned model weight matrices  $W_1, \dots, W_K$ , and the hyperparameters  $\rho_1, \dots, \rho_K$ .
  - 2: Initialize  $B \leftarrow 0, \{L_1, \dots, L_K\} \leftarrow 0$
  - 3: **for**  $t = 1$  to  $T$  **do**
  - 4:   Update base matrix  $B \leftarrow \frac{\sum_{i=1}^K (W_i - L_i)}{K}$
  - 5:   **for**  $i = 1$  to  $K$  **do**
  - 6:     Update  $i$  th delta matrix  $L_i \leftarrow$  Singular Value Thresholding of  $W_i - B$  by  $\rho_i$
  - 7:   **end for**
  - 8: **end for**
  - 9: **Output:**  $B, L_1, \dots, L_K$
- 

**Initialization** To start with, we initialize the matrix  $B$  and the matrices  $L_i$  to be zero. Please note that in our algorithm, the initialization doesn't have any influence on the algorithm.

**Base Matrix update** At the  $t$  step, in the first phase we update the base matrix  $B$  and fix the delta matrices  $L_1, \dots, L_K$  in the last step. Thus, the object function for  $B$  is  $\min_B \sum_{i=1}^K \frac{1}{2} \|W_i - B - L_i\|_F^2$ , which has a direct closed-form result

$$B_t = \frac{\sum_{i=1}^K (W_i - L_i)}{K}$$

**Delta Matrix update** In the second phase, we update the delta matrices  $L_1, \dots, L_K$  and fix the updated base matrix  $B_t$ . Thus, the target function is decomposed into  $\min_{L_i} (\frac{1}{2} \|W_i - B - L_i\|_F^2 + \rho_i \|L_i\|_*)$

**Table 1:** Accuracy Comparison: Llama-2-7b model series.

| Model                                           | Evaluation Tasks   | Original Accuracy | Ours         | Direct Delta Decomposition | BitDelta (with calibration) | 4 bit Quantization |
|-------------------------------------------------|--------------------|-------------------|--------------|----------------------------|-----------------------------|--------------------|
| <i>meta-llama/Llama-2-7b-hf</i>                 | MMLU               | 45.91             | <b>49.79</b> | 45.91                      | 45.91                       | 42.88              |
| <i>meta-llama/Llama-2-7b-chat-hf</i>            | MMLU               | 47.41             | <b>50.48</b> | 48.11                      | 47.29                       | 45.71              |
|                                                 | TruthfulQA         | 45.32             | <b>42.72</b> | <b>48.38</b>               | 44.62                       | 45.00              |
| <i>lmsys/vicuna-7b-v1.5</i>                     | MMLU               | 49.95             | <b>50.95</b> | 48.80                      | 49.83                       | 47.69              |
|                                                 | TruthfulQA         | 50.36             | 47.26        | 48.35                      | 49.95                       | <b>50.43</b>       |
| <i>clibrain/Llama-2-7b-ft-instruct-es</i>       | MMMLU_ES           | 39.38             | <b>42.76</b> | 39.32                      | 39.52                       | 36.95              |
| <i>hfl/chinese-llama-2-7b</i>                   | CEVAL              | 36.40             | <b>39.90</b> | 36.40                      | 29.72                       | 32.39              |
| <i>nlpie/Llama2-MedTuned-7b</i>                 | MedQA              | 39.59             | <b>41.95</b> | 39.28                      | 40.14                       | 38.41              |
| <i>elyza/ELYZA-japanese-Llama-2-7b-instruct</i> | JNLI               | 44.69             | <b>50.43</b> | 46.93                      | 44.01                       | –                  |
| <i>shan2003/llama-2-7b-legal-laws</i>           | Law_stack_exchange | 23.04             | <b>63.01</b> | 55.16                      | 61.44                       | 55.96              |

**Table 2:** Accuracy Comparison: Llama-2-13b model series.

| Model                                            | Evaluation Tasks | Original Accuracy | Ours         | Direct Delta Decomposition | BitDelta | 4 bit Quantization |
|--------------------------------------------------|------------------|-------------------|--------------|----------------------------|----------|--------------------|
| <i>meta-llama/Llama-2-13b-hf</i>                 | MMLU             | 54.91             | <b>56.86</b> | 54.91                      | 54.91    | 53.59              |
| <i>meta-llama/Llama-2-13b-chat-hf</i>            | MMLU             | 53.58             | <b>56.67</b> | 54.49                      | 53.77    | 52.33              |
|                                                  | TruthfulQA       | 43.96             | 42.94        | <b>44.79</b>               | 42.85    | 41.38              |
| <i>lmsys/vicuna-13b-v1.5</i>                     | MMLU             | 55.71             | <b>57.53</b> | 56.78                      | 55.74    | 54.74              |
|                                                  | TruthfulQA       | <b>50.86</b>      | 45.37        | 48.13                      | 50.28    | 47.96              |
| <i>clibrain/Llama-2-13b-ft-instruct-es</i>       | MMMLU_ES         | 46.14             | <b>47.41</b> | 46.17                      | 45.90    | 44.50              |
| <i>hfl/chinese-llama-2-13b</i>                   | CEVAL            | 41.83             | <b>43.54</b> | 41.83                      | 37.59    | 40.34              |
| <i>nlpie/Llama2-MedTuned-13b</i>                 | MedQA            | 42.18             | <b>45.95</b> | 43.91                      | 42.34    | 40.85              |
| <i>elyza/ELYZA-japanese-Llama-2-13b-instruct</i> | JNLI             | 55.76             | <b>59.16</b> | 53.52                      | 52.54    | –                  |

for each delta matrix  $L_i$ . The optimized result can be achieved using Singular Value Thresholding (SVT) techniques [3].

The detailed SVT process is described as follows. First, a singular value decomposition (SVD) is applied to the residual matrix  $W - B$ , yielding  $W - B = USV^T$ . Next, a thresholding operation is applied to the singular values in  $S$ , effectively retaining only those exceeding the value  $\rho_i$ . The resulting matrix of thresholded singular values is denoted as  $\hat{S}$ . Finally, the solution  $L_i$  is constructed as  $L_i = U\hat{S}V^T$ .

The entire process is summarized in Algorithm 1. Note that convergence may occur before reaching the maximum iteration count  $T$ . Specifically, if the updated matrices  $B$  and  $L_1, \dots, L_K$  do not significantly change between consecutive iterations, we can consider the algorithm to have converged. In our practical experience, we find that convergence typically occurs within a small number of iterations.

Moreover, we can claim that the practice of the baseline approach outlined in Section 3.2 is equivalent to solving a special case of our optimization problem 1, given the illustration of SVT techniques. More precisely, the baseline approach is akin to minimizing the following objective:

$$\min_{L_0, \dots, L_K} \sum_{i=1}^K \left( \frac{1}{2} \|W_i - P - L_i\|_F^2 + \phi_i \|L_i\|_* \right) \quad (2)$$

where  $\phi_i$  is implicitly chosen to be the value of the delta matrix  $D_i$ . This choice effectively prunes the smaller singular values by  $\phi_i$ . In essence, the baseline approach can be viewed as a constrained instance of our more general optimization framework 1.

### 3.4 Theoretical analysis

In this section, we provide a rigorous convergence proof and analysis for the proposed optimization algorithm, based on Block Coordinate Descent (BCD). Consider the optimization problem:

$$\min_{B, \{L_i\}} \frac{1}{2} \sum_{i=1}^K \|W_i - B - L_i\|_F^2 + \lambda \sum_{i=1}^K \|L_i\|_*,$$

where  $\|\cdot\|_*$  denotes the nuclear norm. We define the objective function as follows:

$$F(B, L) = f(B, L) + g(L),$$

with

$$f(B, L) = \frac{1}{2} \sum_{i=1}^K \|W_i - B - L_i\|_F^2, \quad g(L) = \lambda \sum_{i=1}^K \|L_i\|_*.$$

**Lemma 1.** For any matrix  $X = U\Sigma V^T$  with singular values  $\sigma_j$ , the singular value thresholding operator minimizes the function  $\frac{1}{2} \|X - Y\|_F^2 + \rho \|X\|_*$ .

*Proof.* This lemma follows directly from Theorem 2.1 in Cai et al. (2010)[3].

**Theorem 1. (Global Convergence)** Algorithm 1 converges to a global minimum.

*Proof.* First, observe that  $F(B, L)$  is convex and continuous, ensuring the existence of a global minimizer. Each iteration of the algorithm consists of exact minimizations with respect to each block while keeping others fixed, thus implying a non-increasing sequence  $\{F(B^{(t)}, L^{(t)})\}$ . Specifically,

$$F(B^{(t+1)}, L^{(t+1)}) \leq F(B^{(t)}, L^{(t+1)}) \leq F(B^{(t)}, L^{(t)}).$$

Since  $F(B, L)$  is bounded below, the sequence  $F(B^{(t)}, L^{(t)})$  must converge.

Consider any accumulation point  $(\bar{B}, \bar{L})$  of the sequence  $(B^{(t)}, L^{(t)})$ . By optimality conditions, it satisfies the KKT equations:

$$0 \in \bar{L}_i - (W_i - \bar{B}) + \lambda \partial \|\bar{L}_i\|_*, \quad 0 = K\bar{B} - \sum_{i=1}^K (W_i - \bar{L}_i).$$

The convexity ensures these conditions characterize the global optimum uniquely in  $B$ , guaranteeing that  $(B^{(t)}, L^{(t)})$  converges to the global minimizer.  $\square$

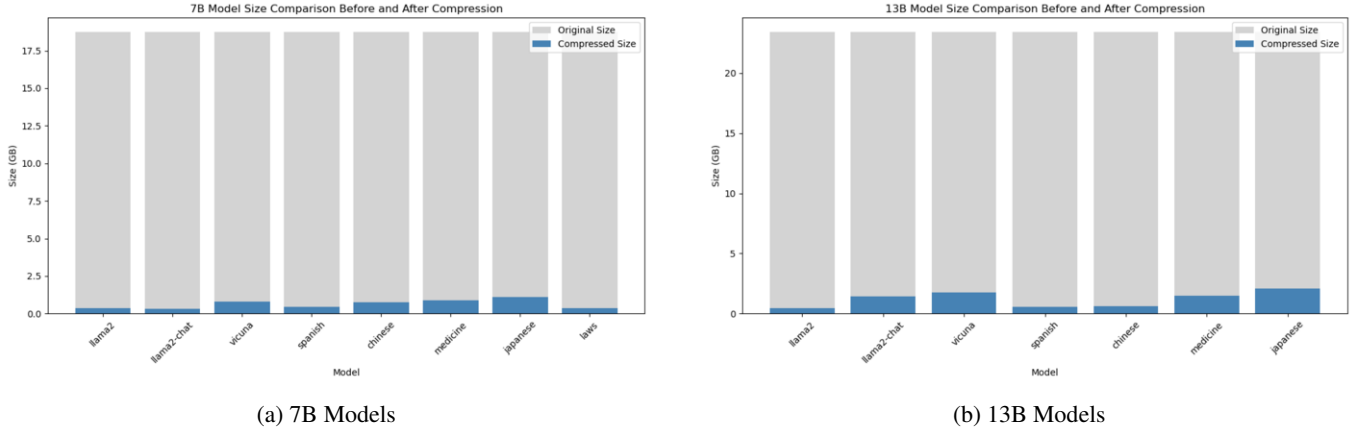


Figure 3: Model Size Comparison Before and After LoRA Decomposition

**Theorem 2. (Convergence Rate)** *The sequence  $\{F(B^{(t)}, L^{(t)})\}$  converges with a sublinear rate of  $O(1/t)$ .*

*Proof.* Since  $F(B, L)$  has a Lipschitz continuous gradient in the variable  $B$  with Lipschitz constant  $K$ , and the updates for  $L_i$  are exact proximal steps of a convex nonsmooth regularizer, standard results from proximal gradient and coordinate descent methods (Tseng, 2001) imply:

$$F(B^{(t)}, L^{(t)}) - F(B^*, L^*) \leq \frac{C}{t},$$

where  $C$  is a constant dependent on the initial point  $(B^{(0)}, L^{(0)})$  and the optimal point  $(B^*, L^*)$ . Hence, the convergence rate of the objective function to its global minimum is  $O(1/t)$ .  $\square$

## 4 Experiment

In this section, we empirically validate our method LoRA Decompose to demonstrate its effect on memory reduction and the influence on model accuracy of downstream tasks.

### 4.1 Setup

**Baselines.** Our primary baselines are the original fine-tuned models without decomposition. We also compare with Direct Delta Compressed models, Bitdelta quantized models and 4-bit quantized models.

**Datasets and Models.** We benchmark a range of models fine-tuned based on the Llama-2 pretrained model, each targeting at different downstream tasks. These include Llama2-chat and Vicuna, and also some models specifically fine-tuned for tasks in various languages such as Chinese, Spanish, and Japanese, as well as specialized fields like medicine and law. Furthermore, we have explored the impact of varying model parameters within the same domain, examining configurations like 7B and 13B.

We evaluate different models on different tasks: 5-shot MMLU for Llama2, Llama2-chat, Vicuna; 5-shot m\_mmlu\_es for Spanish model; 5-shot ceval\_valid for Chinese model; 5-shot jnli for Japanese model; 5-shot medqa\_4options for medical model; 5-shot law\_stack\_exchange for law model. We mostly use lm-evaluation-harness to evaluate on these tasks.

We concentrate on evaluation tasks that are highly relevant to task-specific performance, rather than those related to the model’s general performance. In our main experiments, all eight models we chose are collaboratively compressed together.

As shown in Figure 3, we can significantly compress the storage required compared to storing the full size of all the models.

With the surprising performance on memory reduction, we also check how these models perform on their downstream tasks. The result of 7B models is shown in Table 1 and 13B models in Table 2. We find that our method does quite well on these downstream metrics, generally better than other compression methods.

We note that our method’s accuracy is better than the original accuracy. We attribute this to the knowledge gain merged from their companion models during our compression method.

### 4.2 Ablation

**The effect of number of models** To explore how the number of models  $K$  affects the model performance, we design an experiment to increase  $K$  progressively, i.e., the number of models involved in the collaborative compression.

As shown in Figure 4, we choose Llama2-7B on MMLU task in the left and MedTuned-7B on MedQA task. As the model number  $K$  increases, there is a general trend that the model gains better accuracy on their downstream tasks. This further illustrates the phenomenon that our method’s accuracy may exceed the original accuracy during our collaborative compression process.

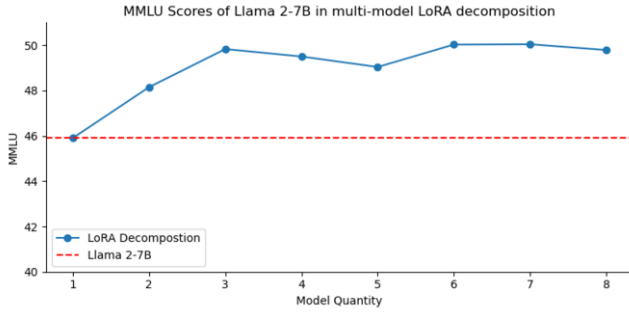
**Compression details within models** We provide compression details for each layer of the model. In Figure 5, we choose both Vicuna-7B model and Vicuna-13B model. Generally, the ranks in FFN layers are apparently larger than in Attention. They are more difficult to compress since the size of the matrices is larger and they are less likely to fall into the low-dimensional structure as their counterparts in Attention layers.

## 5 Related Work

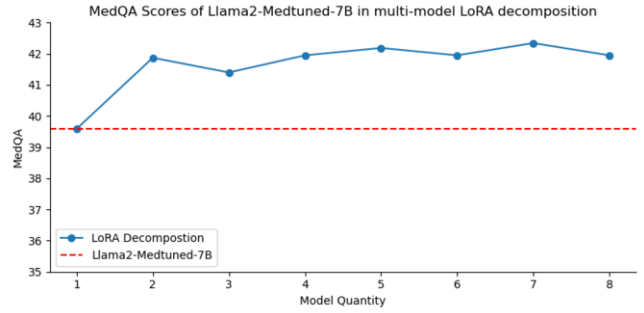
**Model Fine-tuning.** Fine-tuning tailors pretrained models into dataset specific or task-specific models. The methods of which can be divided into two categories. Full parameter fine-tuning, the straightforward one, involves a lot of computation and memory resources. Parameter Efficient Fine-tuning aims at reducing the resource burden while keeping up the model performance.

The challenge of full parameter fine-tuning lies in its nature of high resource requirements, primarily constrained by memory. [24, 17] minimize gradient memory footprint.

Parameter Efficient Fine-tuning aims at reducing the memory and computation resources during fine-tuning. Among many PEFT methods, LoRA [11] is the most popular one due to its simplicity and effectiveness. In [7], the authors of LoRA [11] suggests that LoRA

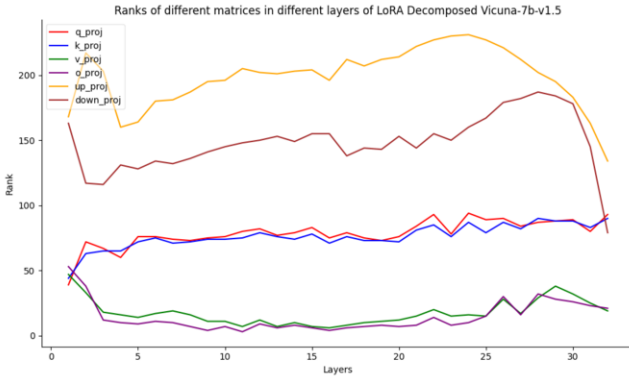


(a) Llama2-7B

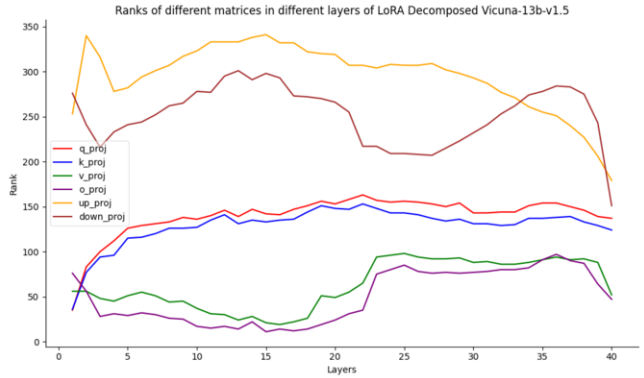


(b) Llama2-MedTuned-7B

Figure 4: Performance of Models in multi-model LoRA Decomposition.



(a) Vicuna-7b-v1.5



(b) Vicuna-13b-v1.5

Figure 5: Ranks of matrices in different layers after LoRA Decomposition

modules can be added to FFN as well. QLoRA aims at further reducing the fine-tuning resources by trading off the fine-tuned model accuracy with quantized base model. LoftQ mitigates this accuracy tradeoff by t, through the view that it would be beneficial to further bridge the weight deviation gap.

We take the similar point of view by improving the gap comparing to the baseline, i.e. delta decomposition described in Section 3

**Delta Compression** BitDelta[14] proposes compressing the delta weights between fine-tuned and pretrained models using only one-bit quantization. However, it requires post-training calibration on representative datasets, which introduces additional computational overhead and may result in suboptimal generalization across diverse downstream tasks. Delta-CoMe[19] addresses the accuracy degradation caused by direct low-rank compression by incorporating auxiliary low-bit quantization. Despite this improvement, its compression rate remains limited to 16x. Therefore, achieving both higher compression ratios and competitive model performance remains an open challenge.

**Model Inference.** To inference multiple models in parallel, bitdelta compress the delta weight between pretrained model and finetuned model into one bit quantization value.

Both SLoRA[22] and Punica[4] foresee the opportunity to efficiently serve multiple LoRA finetuned models at the same time. They designed custom kernels to pack LoRA matmul computation into one for multiple models.

**LoRA expressiveness** In [20], the authors make a comparison between full parameter fine-tuning and LoRA fine-tuning. Their results show that LoRA fine-tuning can have similar results with full parameter fine-tuning.

## 6 Discussion

**Conclusion** In this paper, we present LoRA Decompose, a novel multi-model compression framework for discovering the low-rank structure underlying a set of fine-tuned models. Our work was motivated by the observation that using a pretrained model as a shared base for compression might not be optimal. We delved into the analysis of delta-based compression methods, which led us to formulate the problem of finding an optimal shared base and low-rank delta matrices as a constrained optimization problem. We then devised a simple yet effective block coordinate descent algorithm to solve this problem. Our approach demonstrates the potential for significant model compression while maintaining performance, offering a promising avenue for efficient deployment of large language models in resource-constrained environments.

**Limitations** Our is not a universal compression algorithm that can both decompose any fine-tuned model into fairly low-rank result and keep up the accuracy performance. For example, we fail to compress the codellama 7b model together with the pretrain model llama-2 7b. Our hypothesis is that this observation is related to fine-tuning dataset. We think either the size of the fine-tuning dataset or the deviation of the fine-tuning dataset to the pretraining dataset may influence this observation. We were unable to conduct such experiments due to the lack of large scale dataset and the lack of computing resources. This hypothesis can be done for further research.

**Impact** In this paper, we justify the user case where multiple model inference is within reach while there is still lack of research optimizing this scenario. To date, our work is the first research that targets collaboratively compress a set of models. Also, we are inference-friendly thanks to Punica and SLoRA.

## Acknowledgements

This work was supported by the National Key Research and Development Program of China under Grant 2023YFB4405102. (\*Corresponding author: Ningyi Xu).

## References

- [1] M. Azure. New models added to the phi-3 family available on microsoft azure, 2024. URL <https://azure.microsoft.com/en-us/blog/new-models-added-to-the-phi-3-family-available-on-microsoft-azure/>. Accessed: 2024-05-22.
- [2] P. Belcak, G. Heinrich, S. Diao, Y. Fu, X. Dong, S. Muralidharan, Y. C. Lin, and P. Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.
- [3] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4): 1956–1982, 2010.
- [4] L. Chen, Z. Ye, Y. Wu, D. Zhuo, L. Ceze, and A. Krishnamurthy. Punica: Multi-tenant lora serving. *arXiv preprint arXiv:2310.18547*, 2023.
- [5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [6] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.
- [7] V. Fomenko, H. Yu, J. Lee, S. Hsieh, and W. Chen. A note on lora. *arXiv preprint arXiv:2404.05086*, 2024.
- [8] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.
- [9] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [10] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations, ICLR*, 2024.
- [11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [12] Y. Li, Y. Yu, C. Liang, N. Karampatziakis, P. He, W. Chen, and T. Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2023.
- [13] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.
- [14] J. Liu, G. Xiao, K. Li, J. D. Lee, S. Han, T. Dao, and T. Cai. Bitdelta: Your fine-tune may only be worth one bit. *Advances in Neural Information Processing Systems*, 37:13579–13600, 2024.
- [15] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.
- [16] H. Luo, Q. Sun, C. Xu, P. Zhao, J. Lou, C. Tao, X. Geng, Q. Lin, S. Chen, and D. Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.
- [17] K. Lv, Y. Yang, T. Liu, Q. Gao, Q. Guo, and X. Qiu. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*, 2023.
- [18] L. Peng and R. Vidal. Block coordinate descent on smooth manifolds. *arXiv preprint arXiv:2305.14744*, 2023.
- [19] B. Ping, S. Wang, H. Wang, X. Han, Y. Xu, Y. Yan, Y. Chen, B. Chang, Z. Liu, and M. Sun. Delta-come: Training-free delta-compression with mixed-precision for large language models. *arXiv preprint arXiv:2406.08903*, 2024.
- [20] O. Razuvayevskaya, B. Wu, J. A. Leite, F. Heppell, I. Srba, C. Scarton, K. Bontcheva, and X. Song. Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification. *Plos one*, 19(5):e0301738, 2024.
- [21] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.
- [22] Y. Sheng, S. Cao, D. Li, C. Hooper, N. Lee, S. Yang, C. Chou, B. Zhu, L. Zheng, K. Keutzer, et al. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*, 2023.
- [23] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.
- [24] Y. Zhang, Y. Han, S. Cao, G. Dai, Y. Miao, T. Cao, F. Yang, and N. Xu. Adam accumulation to reduce memory footprints of both activations and gradients for large-scale dnn training. *arXiv preprint arXiv:2305.19982*, 2023.