

# IMPART: Importance-Aware Delta-Sparsification for Improved Model Compression and Merging in LLMs

Yan Yang<sup>1\*</sup>, Yixia Li<sup>3\*</sup>, Hongru Wang<sup>2</sup>, Xuetao Wei<sup>3</sup>  
James Jianqiao Yu<sup>4</sup>, Yun Chen<sup>1†</sup>, Guanhua Chen<sup>3†</sup>

<sup>1</sup>Shanghai University of Finance and Economics, <sup>2</sup>The Chinese University of Hong Kong  
<sup>3</sup>Southern University of Science and Technology, <sup>4</sup>Harbin Institute of Technology (Shenzhen)  
yanyang@stu.sufe.edu.cn, liyixia@me.com  
yunchen@sufe.edu.cn, chengh3@sustech.edu.cn

## Abstract

With the proliferation of task-specific large language models, delta compression has emerged as a method to mitigate the resource challenges of deploying numerous such models by effectively compressing the delta model parameters. Previous delta-sparsification methods either remove parameters randomly or truncate singular vectors directly after singular value decomposition (SVD). However, these methods either disregard parameter importance entirely or evaluate it with too coarse a granularity. In this work, we introduce IMPART, a novel importance-aware delta sparsification approach. Leveraging SVD, it dynamically adjusts sparsity ratios of different singular vectors based on their importance, effectively retaining crucial task-specific knowledge even at high sparsity ratios. Experiments show that IMPART achieves state-of-the-art delta sparsification performance, demonstrating  $2\times$  higher compression ratio than baselines at the same performance level. When integrated with existing methods, IMPART sets a new state-of-the-art on delta quantization and model merging.

## 1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across diverse knowledge-intensive (Yang et al., 2024; Abdin et al., 2024) and reasoning-intensive (DeepSeek-AI et al., 2025; Kimi Team et al., 2025) tasks through post-training. Different users fine-tune the widely applicable open-sourced base LLMs such as LLaMA (Grattafiori et al., 2024) and DeepSeek (DeepSeek-AI et al., 2024) with customized datasets for specific downstream tasks. However, maintaining separate fine-tuned models for each user or downstream task poses significant resource challenges (Ryu et al., 2023; Yao et al., 2024), particularly in storage and deployment costs. The challenges have

attracted increased interest within the community in efficient model compression techniques that can preserve task-specific knowledge while reducing resource requirements.

Recent approaches (Yu et al., 2024; Liu et al., 2024; Ping et al., 2024) propose to address this challenge by delta compression, which aims to compress the difference between the fine-tuned parameters and the base model parameters (i.e., delta parameters) by quantization or sparsification. Previous sparsification-based methods (Isik et al., 2023; Yao et al., 2024; Yu et al., 2024) sparsify the delta parameters by randomly setting partial weight entries to zero or truncating singular vectors directly after singular value decomposition (SVD). However, these methods fail to produce satisfactory results, particularly on challenging specialized tasks like math reasoning or code generation, as they inadvertently discard important parameters as the sparsification ratio increases.

In this work, we propose IMPART (**I**mportance-**A**ware **D**elta-**S**parsification), a novel effective sparsification-based delta compression approach even at high sparsity ratios. The IMPART framework is motivated by the observations that singular vectors associated with larger singular values encode more important task-specific information (Ping et al., 2024; Sharma et al., 2024; Ryu et al., 2023). Building on this insight, IMPART proposes an adaptive sparsification mechanism that assigns different sparsity ratios on the singular vectors based on the corresponding singular values, ensuring the preservation of critical task-specific knowledge. Based on our theoretical analysis, the parameters of the sparsified singular vector are then re-scaled to ensure the performance is maintained. The IMPART framework can be applied to delta quantization or model merging tasks by integrating it with existing approaches, thereby supporting a higher compression ratio.

Extensive experiments on LLM sparsification

\* Equal Contribution.

† Corresponding Authors.

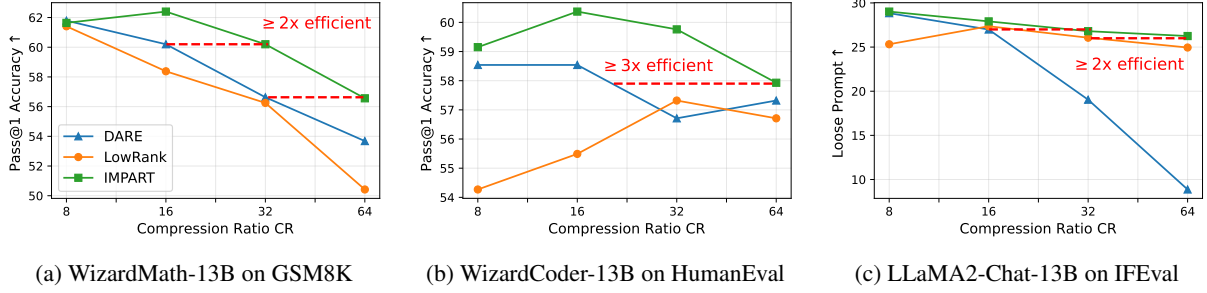


Figure 1: Comparative evaluation of IMPART against state-of-the-art sparsification methods across mathematical reasoning, code generation, and chat tasks. IMPART consistently outperforms baselines across various tasks while maintaining high sparsity ratios (more detailed discussions are in Section 6.3).

across three diverse tasks with various backbones demonstrate the effectiveness of our method. As shown in Figure 1, IMPART demonstrates  $2\times$  higher compression ratio than baselines at the same performance level, retaining 95.8% of the fine-tuned model’s performance at a compression ratio of 16 (93.75% sparsity). Additional experiments on integration with quantization and model merging further validate IMPART’s versatility, making it a practical solution for deploying numerous fine-tuned language models in resource-constrained environments.<sup>1</sup>

## 2 Preliminaries

**Delta Compression** Delta parameters are the differences between the parameters of a fine-tuned LLM and its corresponding base LLM. In scenarios such as multi-tenant serving, where a large number of LLMs fine-tuned from the same base model are deployed to meet various and complicated user requirements, using  $N$  sets of delta parameters in conjunction with the shared backbone can eliminate the need for  $N$  full fine-tuned models. Delta compression aims to compress these delta parameters by sparsification (Yu et al., 2024; Yao et al., 2024), quantization (Isik et al., 2023; Liu et al., 2024), or merging (Wortsman et al., 2022; Yadav et al., 2023) to reduce the overall number of parameters. Thereby delta compression decreases both storage requirements and GPU memory utilization in scenarios involving multiple fine-tuned models.

**Delta Parameter Decomposition** Given a delta parameter  $\Delta W \in \mathbb{R}^{m \times n}$ , its singular value decomposition (SVD) can be expressed as  $\Delta W = U\Sigma V^\top$ , where  $U \in \mathbb{R}^{m \times m}$ ,  $V \in \mathbb{R}^{n \times n}$ , and

$\Sigma \in \mathbb{R}^{m \times n}$  contains the singular values in descending order. Assuming  $n \leq m$  for simplicity, we can reformulate the SVD as:

$$\Delta W = U\Sigma V^\top = \sum_{i=1}^n \sigma_i^\downarrow U_i V_i^\top, \quad (1)$$

where  $U_i$  and  $V_i$  denote the  $i$ -th columns of  $U$  and  $V$ , respectively, and  $\sigma_i^\downarrow$  represent the singular values ordered in descending magnitude.

We formally define the **sparsity ratio (SR)**  $\alpha \in [0, 1]$  as 1 minus the ratio of the number of non-zero parameters in the sparsified delta parameters to the total number of delta parameters. The corresponding **compression ratio** is given by  $\text{CR} = 1/(1 - \alpha)$ . For instance, a compression ratio of 32 corresponds to  $\alpha \approx 0.97$  (97% sparsity), yielding a 32-fold reduction in storage requirements. Through subsequent quantization, the sparse model can achieve even higher compression ratios, denoted as  $\text{CR}_{\text{qt}}$  to differentiate from the sparsification-only compression ratio CR.

## 3 Methodology

### 3.1 Importance-Aware Sparsification

As shown in Figure 2, IMPART is an importance-aware sparsification method that adaptively allocates sparsity ratios to singular vectors based on their singular values’ magnitude. Larger singular values indicate greater importance of the corresponding singular vectors (Wang et al., 2025b; Gao et al., 2024), we thus assign them a smaller sparsity ratio. Conversely, singular vectors associated with smaller singular values will be given a larger sparsity ratio. Different from previous random sparsification methods like DARE (Yu et al., 2024) and low-rank approximation methods (Ryu et al., 2023; Saha et al., 2024), our method fully

<sup>1</sup>Our code are publicly available at <https://github.com/sustech-nlp/ImPart>.

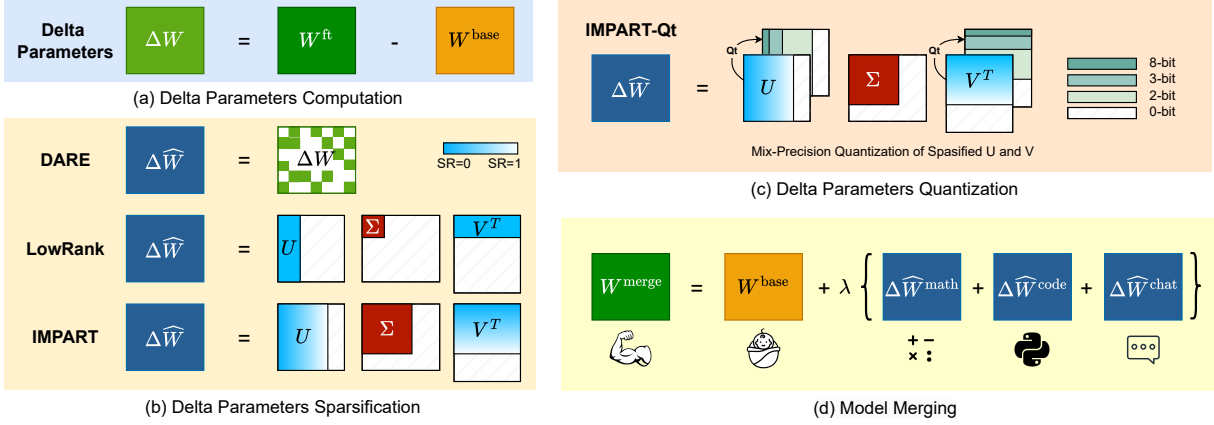


Figure 2: Overview of IMPART. (a) Delta parameters computation by subtracting the base model from the fine-tuned model. (b) Comparison of delta parameters sparsification methods: DARE randomly drops delta parameters, LowRank sparsifies with low-rank approximation, and IMPART adaptively sparsifies singular vectors. (c) Further apply mixed-precision quantization on sparse singular vectors to achieve higher compression ratios. (d) Model merging by combining sparsified delta parameters to build a unified multi-task model.

considers the importance of parameters in the SVD space, allowing for improved sparsity ratios while enhancing the performance of the sparse model.

Specifically, given  $\Delta W$  with singular values  $\{\sigma_k\}_{k=1}^n$ , we allocate a pre-defined sparsity ratio  $p_k$  (see more details in Section 3.2) for the  $k$ -th singular vector pair ( $U_k$  and  $V_k$ ), ensuring the average sparsity ratio across all singular vectors meets the target overall sparsity ratio  $\alpha$ . Inspired by the drop-and-rescale sparsification strategy in DARE (Yu et al., 2024), we then sample independent Bernoulli random variables  $\xi_k^i$  and  $\eta_k^j$  to randomly mask the singular vectors  $U_{ik}$  (Equation 4) and  $V_{kj}$  (Equation 5) according to their corresponding sparsity ratio  $p_k$ . To approximate the original singular vector, we apply a rescaling coefficient of  $1/(1-p_k)$  to the remaining parameters (see more discussions on how to select the coefficient in Section 3.3). The delta parameter is then reconstructed with sparsified singular vectors (Equation 6).

$$\xi_k^i \sim \text{Bernoulli}(1-p_k), i \in [1, m], k \in [1, n] \quad (2)$$

$$\eta_k^j \sim \text{Bernoulli}(1-p_k), k \in [1, n], j \in [1, n] \quad (3)$$

$$\hat{U}_{ik} = U_{ik} \frac{\xi_k^i}{1-p_k}, i \in [1, m], k \in [1, n] \quad (4)$$

$$\hat{V}_{kj} = V_{kj} \frac{\eta_k^j}{1-p_k}, k \in [1, n], j \in [1, n] \quad (5)$$

$$\Delta \hat{W} = \hat{U} \cdot \Sigma \cdot \hat{V}^T \quad (6)$$

### 3.2 Strategy for Sparsity Ratio Allocation

Given a pre-defined overall sparsity ratio  $\alpha$  and  $\Delta W$  with singular values  $\{\sigma_k\}_{k=1}^n$ , we allocate

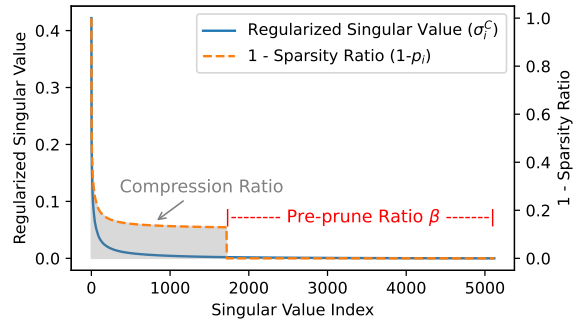


Figure 3: Importance-aware delta-sparsification adaptively sets sparsity ratios based on singular values, ensuring critical information retention. IMPART first pre-prunes small singular components and then allocates sparsity budget based on regularized singular values.

sparsity ratio  $p_k$  to each singular vector pair ( $U_k$  and  $V_k$ ) based on the singular value  $\sigma_k$ , as shown in Figure 3:

$$p_k = \begin{cases} 1 & \text{if } k > \lfloor n \cdot (1 - \beta) \rfloor \\ \left(1 - \left(\frac{\sigma_k}{\sigma_1}\right)^C\right) \cdot \gamma & \text{otherwise} \end{cases} \quad (7)$$

where  $\beta$  and  $C$  are hyperparameters selected with a validation set, and  $\gamma$  is a scaling factor calculated for each  $\Delta W$  to ensure the overall sparsity ratio  $\alpha$  is met. Our allocation strategy is designed following two key insights: 1) Previous works show that directly removing the smallest singular components can achieve performance comparable to or even better than using the full set of parameters, due to the long tail distribution of singular values (Ping et al., 2024; Sharma et al., 2024; Ryu et al.,

2023). This observation motivates us to design a pre-pruning ratio, denoted as  $\beta$ , which aims to directly remove these long-tail singular components. 2) For the rest singular components, we allocate the sparsity ratio based on the regularized singular value, with  $C$  serving as a regularization hyperparameter. In practice, it is possible that we cannot achieve the target sparsity ratio  $\alpha$  by simply scaling  $\gamma$  as  $p_k$  is constrained to be less than 1. We address this issue by shifting the boundary of the piecewise function to the left, thereby attaining the desired sparsity ratio. See Algorithm 1 in Appendix A for more details.<sup>2</sup>

### 3.3 Theoretical Analysis

We provide theoretical proof that the expectation of the reconstructed output matches the original output. Given a fine-tuned weight  $W^{\text{ft}} \in \mathbb{R}^{m \times n}$  and an input  $X \in \mathbb{R}^n$ , the expectation of the  $i$ -th ( $1 \leq i \leq m$ ) dimension of the hidden state  $h \in \mathbb{R}^m$  is computed as:

$$\begin{aligned} \mathbb{E}[h_i] &= \mathbb{E}\left[\sum_j W_{ij}^{\text{ft}} X_j\right] \\ &= \mathbb{E}\left[\sum_j W_{ij}^{\text{base}} X_j\right] + \mathbb{E}\left[\sum_j \Delta W_{ij} X_j\right] \\ &= \sum_j W_{ij}^{\text{base}} X_j + \sum_j \Delta W_{ij} X_j \\ &= h_i^{\text{base}} + \sum_j \sum_k \sigma_k U_{ik} V_{kj} X_j, \end{aligned} \quad (8)$$

where  $h_i^{\text{base}}$  is the  $i$ -th dimension of the base model output. Without loss of generality, we assume that the bias term is zero. As IMPART randomly drops the  $k$ -th column of  $U$  and  $V$  independently with a sparsity ratio of  $p_k$ , the expectation of the reconstructed hidden state  $\widehat{h}_i$  is then computed as:

$$\begin{aligned} \mathbb{E}[\widehat{h}_i] &= \mathbb{E}[\widehat{W}^{\text{ft}} X_j] \\ &= \mathbb{E}\left[\sum_j W_{ij}^{\text{base}} X_j\right] + \mathbb{E}\left[\sum_j \Delta \widehat{W}_{ij} X_j\right] \\ &= h_i^{\text{base}} + \mathbb{E}\left[\sum_j \sum_k \sigma_k \widehat{U}_{ik} \widehat{V}_{kj}\right] X_j \\ &= h_i^{\text{base}} + \sum_j \sum_k \sigma_k \mathbb{E}[\widehat{U}_{ik}] \mathbb{E}[\widehat{V}_{kj}] X_j \\ &= h_i^{\text{base}} + \sum_j \sum_k \sigma_k [\theta \cdot (1 - p_k) \cdot U_{ik} + 0 \cdot p_k \cdot U_{ik}] \\ &\quad [\zeta \cdot (1 - p_k) \cdot V_{kj} + 0 \cdot p_k \cdot V_{kj}] X_j \\ &= h_i^{\text{base}} + \sum_j \sum_k \sigma_k [\theta \cdot (1 - p_k) \cdot U_{ik}] [\zeta \cdot (1 - p_k) \cdot V_{kj}] X_j. \end{aligned} \quad (9)$$

<sup>2</sup>To achieve an overall sparsity ratio of  $\alpha$ , the sparsity ratios of  $U$  and  $V$  are approximately  $(1 + \alpha)/2$  for a square matrix.

By setting the rescaling coefficient  $\theta = \zeta = 1/(1 - p_k)$ , we ensure that the reconstructed embedding approximates the origin. We give empirical evidence in Section 6.1 to support this theoretical analysis, where the removal of the rescaling factor leads to significant performance degradation.

## 4 Applications of IMPART

### 4.1 Delta Parameter Quantization

Previous work has demonstrated that delta parameters can be effectively compressed from 16-bits to 1-bit using low-bit quantization methods such as BitDelta (Liu et al., 2024) and Delta-CoMe (Ping et al., 2024). In this section, we enhance the compression ratio without sacrificing performance by combining IMPART, a delta parameter sparsification technique, with delta parameter quantization methods. Since IMPART is based on SVD, we integrate it with Delta-CoMe, a state-of-the-art mixed-precision quantization method that also operates in the SVD space. It is important to note that DARE cannot be integrated with Delta-CoMe, as SVD will break the sparse weight matrix created by DARE.

**Delta-CoMe** Delta-CoMe is a mixed-precision delta parameter quantization method. Instead of directly quantizing  $\Delta W$ , it first decomposes the delta parameter with the SVD method and then quantizes all the singular vectors using the GPTQ method (Frantar et al., 2023). During the process of GPTQ quantization, singular vectors corresponding to larger singular values are allocated with larger bit-widths, due to their greater impact on the approximation of delta weights.

**IMPART-QT** The IMPART-QT framework is a highly efficient mixed-precision delta compression method that combines the strengths of IMPART and Delta-CoMe methods. To integrate IMPART with Delta-CoMe, we first use IMPART to sparsify the delta parameter, and then apply Delta-CoMe to the sparsified singular vectors. However, this is not trivial. We address potential issues such as the quantization of sparse singular matrices by Delta-CoMe, the allocation of compression ratios for sparsification and quantization, and other related concerns in Appendix B.

### 4.2 Model Merging

Model merging aims to merge multiple task-specific fine-tuned models into a single model with diverse abilities (Ilharco et al., 2023; Yadav et al.,

2023). Recently, it has attracted the attention of the research community for its cost-effectiveness, knowledge-sharing potential, and space efficiency. Task Arithmetic (Ilharco et al., 2023, TA) and TIES-Merging (Yadav et al., 2023, TIES) are two commonly used model merging methods (see Appendix C for the details). As a sparsification method, IMPART is able to preserve the abilities of fine-tuned LLM, as long as a small portion of the parameters in singular vectors remain unaffected. This motivates us to employ IMPART before model merging, as IMPART can reduce parameter redundancy in each fine-tuned model before merging, which can potentially mitigate the interference of parameters among multiple fine-tuned models.

Specifically, given  $N$  models fine-tuned on  $N$  distinct tasks from the same base LLM, we first apply IMPART on delta parameters for each fine-tuned model. Then we adopt established model merging methods such as TA and TIES to fuse the derived parameters and obtain the merged single model. The purpose and usage of IMPART are similar to the DARE method (Yu et al., 2024) in model merging, therefore, we also compare our method with DARE in Section 7.2.

## 5 Sparsification Experiments

To evaluate the effectiveness of IMPART, we conduct experiments across three diverse tasks: mathematical problem-solving, code generation, and chat. Our experiments cover various model sizes and backbones, benchmarking IMPART against state-of-the-art methods for model sparsification.

### 5.1 Tasks

**Mathematics** We evaluate on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) using Pass@1 accuracy, focusing on complex mathematical reasoning abilities.

**Code Generation** Performance is assessed on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) using Pass@1 accuracy for natural language to code generation.

**Chat** Models are evaluated using IFEval (Zhou et al., 2023) loose prompt metric for response controllability and AlpacaEval2 (Dubois et al., 2024) length-controlled win rate (LCWR) against GPT4-Turbo baseline, judged by GPT-4o-2024-08-06.

### 5.2 Hyperparameter Selection

For each task, we tune the hyperparameters on the validation set to select the optimal  $\beta$  from {0.6, 0.7, 0.8} and  $C$  from {0.5, 1}. We use SVAMP (Patel et al., 2021) Pass@1, Mercury (Du et al., 2024b) Pass@1 and FollowBench (Jiang et al., 2024) hard satisfaction rate for math, code, and chat tasks as validation sets, respectively.

### 5.3 Models

The model setups are summarized in Table 1. We evaluate IMPART on mainstream fine-tuned models, including WizardMath-13B-V1.0 (Luo et al., 2025) for mathematical problem solving, WizardCoder-13B (Luo et al., 2023) for code generation, and LLaMA2-Chat-13B (Touvron et al., 2023) for chat tasks. To further assess IMPART’s performance across different model sizes and backbones, we also conduct experiments on LLaMA2-Chat-7B (Touvron et al., 2023) and LLaMA3-Instruct-8B (Grattafiori et al., 2024) for chat tasks.

Task	Backbone	Fine-tuned
Math	LLaMA2-13B	WizardMath-13B-V1.0
Code	Codellama-13B	WizardCoder-13B
Chat	LLaMA2-13B	LLaMA2-Chat-13B
Chat	LLaMA2-7B	LLaMA2-Chat-7B
Chat	LLaMA3-8B	LLaMA3-Instruct-8B

Table 1: Selected backbones and fine-tuned LLMs for the examined tasks.

### 5.4 Baselines

**DARE** We compare against DARE (Yu et al., 2024), a delta compression method through random delta parameter sparsification.

**LowRank** We implement a simple SVD-based baseline (Ryu et al., 2023) that preserves only the top  $r$  singular values and corresponding singular vectors. This serves as a direct comparison point for evaluating IMPART’s adaptive sparsification mechanism over basic rank truncation.

### 5.5 Results

Table 2 presents the sparsification results for IMPART and baselines across various tasks and backbones. IMPART consistently outperforms both DARE and the LowRank baseline, achieving an average improvement of 4.01 over DARE and 2.11 over the LowRank baseline.

Notably, DARE exhibits significant performance degradation on chat tasks, particularly in AlpacaE-

Methods	CR	WizardMath-13B		WizardCoder-13B		LLaMA2-Chat-13B		LLaMA2-Chat-7B		LLaMA3-Inst-8B		Avg.
		GSM8K	MATH	HumanEval	MBPP	IFEval	AlpacaEval	IFEval	AlpacaEval	IFEval	AlpacaEval	
Backbone <sup>†</sup>	1	17.80	3.90	32.32	62.70	19.04	0.71	20.52	0.10	11.46	0.08	16.86
Fine-tuned <sup>†</sup>	1	63.96	14.10	59.76	67.70	33.64	18.39	31.79	15.63	48.80	32.13	38.59
DARE	32	58.91	<b>11.76</b>	54.27	64.60	24.77	2.27	16.82	0.36	30.50	17.76	28.20
LowRank	32	56.25	7.94	57.32	<b>68.80</b>	26.06	8.45	23.84	5.72	29.39	17.18	30.10
IMPART	32	<b>60.20</b>	10.38	<b>59.76</b>	68.00	<b>26.80</b>	<b>9.88</b>	<b>27.91</b>	<b>7.13</b>	<b>33.27</b>	<b>18.77</b>	<b>32.21</b>

Table 2: Comparison of IMPART and baselines on various tasks across backbones. <sup>†</sup> denotes the uncompressed backbone and fine-tuned models, serving as the reference for sparsification. The best results are highlighted in **bold**.

ID	Ablations	GSM8K	HumanEval	IFEval	Avg.
①	IMPART	<b>60.20</b>	<b>59.76</b>	<b>26.80</b>	<b>48.92</b>
②	w/o Pre-prune	57.92	54.88	26.62	46.47
③	w/o Importance-Aware	0.00	51.83	12.20	21.34
④	w/o Pre-prune w/o I.A.	0.00	20.12	11.83	10.65
⑤	w/o $1/(1-p)$ Rescale	33.21	6.10	22.92	20.74

Table 3: Ablation study on different components of IMPART. I.A. denotes Importance-Aware, and  $1/(1-p)$  rescale refers to the rescale coefficient in Equation 4, 5.

val, where random sparsification leads to repetitive responses and compromises performance on LLaMA2. While the impact on IFEval is less severe due to its rule-based metrics, the overall decline underscores the limitations of random sparsification. In contrast, IMPART’s adaptive strategy mitigates these issues, ensuring better retention of task-relevant knowledge and achieving more reliable results across tasks and backbones.

When comparing IMPART with the LowRank baseline, we observe significant improvements in overall performance and most individual tasks. For instance, with a compression ratio of 32, IMPART only shows a 3.76 decrease on GSM8K, while LowRank exhibits a 7.71 decrease. IMPART maintains performance on HumanEval without any degradation, while LowRank exhibits a 2.44 decrease. These results underscore the effectiveness of IMPART in preserving critical task-specific information and achieving SOTA model sparsification.

## 6 Analyses of IMPART

In this section, we conduct comprehensive analyses of IMPART on three representative tasks: mathematical problem solving (GSM8K with WizardMath-13B), code generation (HumanEval with WizardCoder-13B), and chat (IFEval with LLaMA2-Chat-13B). Unless otherwise specified, we use a compression ratio  $CR = 32$ .

### 6.1 Ablations

To assess the impact of different components of IMPART, we conduct an ablation study on the

pre-pruning parameter  $\beta$ , the importance-aware sparsification strategy, and the effectiveness of the  $1/(1-p)$  rescale, as shown in Table 3.

Our results show that all design components contribute to IMPART. First, pre-pruning long-tail singular vectors with pre-pruning ratio  $\beta$  results in a more effective sparsity allocation strategy, enhancing the performance of the final sparse model by an average of 2.45 (ID 2 vs. ID 1).

We next evaluate our importance-aware sparsification strategy. In IMPART, we adaptively assign sparsity ratios to singular vectors based on their importance values. Comparing this approach against uniform sparsification across unpruned singular vectors (ID 3), we observe that disregarding importance leads to a substantial performance degradation of 27.58 on average (ID 3 vs. ID 1). This deterioration worsens when pre-pruning is removed, with performance dropping by 38.27 (ID 4 vs. ID 1). Most notably, uniform sparsification produces severely degraded outputs with repetition and incoherence, resulting in complete failure (0.00 accuracy) on GSM8K. These findings demonstrate that importance-aware sparsification is crucial for preserving model capabilities.

Finally, we verify the effectiveness of the  $1/(1-p)$  rescale in approximating the original model. When the rescale coefficient is removed from Equation 4 and 5, we observe a significant performance decrease of 28.18 on average (ID 5 vs. ID 1).

### 6.2 Sensitivity Analysis on $\beta$ and $C$

We conduct a comprehensive sensitivity analysis to evaluate the impact of the pre-pruning parameter  $\beta$  and regularization parameter  $C$ . Table 4 presents results across diverse tasks, demonstrating IMPART’s robust performance across various hyperparameter configurations.

Our analysis of the regularization parameter  $C$  reveals task-specific effects. For mathematical reasoning and code generation tasks, maintaining the original singular values ( $C = 1$ ) yields better per-

Methods	CR	$C$	$\beta$	GSM8K	HumanEval	IFEval	Avg.
Backbone <sup>1</sup>	1	-	-	17.80	32.32	19.04	23.05
Fine-tuned <sup>†</sup>	1	-	-	63.96	59.76	33.64	63.81
DARE	32	-	-	58.91	54.27	24.77	45.98
LowRank	32	-	-	56.25	57.32	26.06	46.54
IMPART	32	0.5	0.6	56.48	56.71	<b>27.91</b>	47.03
			0.7	58.07	54.88	25.88	46.28
			0.8	57.62	54.27	<u>26.80</u>	46.23
		1	0.6	<b>60.20</b>	<b>59.76</b>	26.43	<b>48.80</b>
			0.7	58.45	56.71	27.54	47.57
			0.8	58.45	59.15	25.14	47.58

Table 4: Hyperparameter study across different tasks. The best performance is shown in **bold**, and results selected by the validation set are underlined.

formance. In contrast, the chat model benefits from a smaller  $C$  ( $C = 0.5$ ), which reduces the differences between regularized singular values.

Regarding the pre-pruning ratio  $\beta$ , we find that moderate values ( $\beta = 0.6$ ) typically yield optimal results, striking a balance between removing noise and retaining important information. Higher values ( $\beta = 0.7, 0.8$ ) lead to marginally decreased performance, suggesting the loss of important task-specific knowledge during aggressive pre-pruning.

When analyzing the validation set’s selections, we find that it effectively identifies near-optimal hyperparameters for math and code-related tasks but exhibits limitations for chat tasks. For instance, it selects a configuration that achieves 26.80 on IFEval, falling short of the optimal 27.91, likely due to misalignment between validation and test set. Despite this suboptimal configuration, IMPART still outperforms all baselines on chat tasks, highlighting its robustness and effectiveness in model sparsification.

### 6.3 Different Compression Ratios

To demonstrate the flexibility of IMPART, we evaluate performance across varying compression ratios (8 to 64). Table 5 (visualized in Figure 1) demonstrates that IMPART consistently outperforms baseline methods across most settings, with its advantages becoming more pronounced at higher compression ratios. These results validate IMPART’s effectiveness in preserving task-specific knowledge under aggressive sparsification.

## 7 Applications of IMPART

### 7.1 Delta Parameter Quantization

**Setup** We compare IMPART-QT with three baselines: BitDelta<sup>3</sup>, DARE-Qt, and Delta-CoMe, by

<sup>3</sup>As BitDelta is a quantization-based method not suited for sparsification, it was excluded from the sparsification (Sec-

Tasks	Method	8	16	32	64	Avg.
GSM8K	DARE	<b>61.79</b>	60.20	56.63	53.68	58.08
	LowRank	61.41	58.38	56.25	50.42	56.62
	IMPART	61.64	<b>62.40</b>	<b>60.20</b>	<b>56.56</b>	<b>60.20</b>
HumanEval	DARE	58.54	58.54	56.71	57.32	57.78
	LowRank	54.27	55.49	57.32	56.71	55.95
	IMPART	<b>59.15</b>	<b>60.37</b>	<b>59.76</b>	<b>57.93</b>	<b>59.30</b>
IFEval	DARE	28.84	26.99	19.04	8.87	20.94
	LowRank	25.32	27.36	26.06	24.95	25.92
	IMPART	<b>29.02</b>	<b>27.91</b>	<b>26.80</b>	<b>26.25</b>	<b>27.50</b>

Table 5: Performance of IMPART with different compression ratios.

evaluating them with the same model and benchmark setup as in Section 5. We set the target compression ratio  $CR_{qt}$  to 32 for all tasks and models. In line with Delta-CoMe, we employ a triple-precision quantization scheme, assigning 8-bit, 3-bit, and 2-bit precision to distinct singular value groups. See Appendix B.2 for more details.

**Results** Table 6 presents the quantization results for different quantization methods. IMPART-QT achieves the highest overall performance, with an average score of 36.98, surpassing BitDelta by 4.27, DARE-QT by 0.86, and Delta-CoMe by 1.81. These results highlight the effectiveness of IMPART-Qt’s adaptive sparsification strategy in preserving essential task-specific parameters while achieving a high compression ratio. Compared to uncompressed aligned models, IMPART achieves near-lossless performance on math and code tasks. However, there is a relatively greater performance degradation on chat tasks. This suggests that the difficulty of compression varies across different types of tasks. Compared to the sparsification results in Table 2, IMPART-QT achieves significantly better outcomes than IMPART at the same compression ratio of 32. This indicates that for effective compression of the delta parameter, a combination of sparsification and quantization is preferable to using either method alone.

We further present the performance across varying compression ratios, ranging from 16 to 128. As shown in Table 7 (visualized in Figure 4 of Appendix B.5), IMPART-QT consistently outperforms baseline methods across most compression settings.

### 7.2 Model Merging

**Setup** We evaluate model merging on three representative benchmarks for math, code, and chat (Section 5) and model fusion (Section 7.2) experiments.

Methods	CR <sub>qt</sub>	WizardMath-13B		WizardCoder-13B		LLaMA2-Chat-13B		LLaMA2-Chat-7B		LLaMA3-Inst-8B		Avg.
		GSM8K	MATH	HumanEval	MBPP	IFEval	AlpacaEval	IFEval	AlpacaEval	IFEval	AlpacaEval	
Backbone <sup>†</sup>	1	17.80	3.90	32.32	62.70	19.04	0.71	20.52	0.10	11.46	0.08	16.86
Fine-tuned <sup>†</sup>	1	63.96	14.10	59.76	67.70	33.64	18.39	31.79	15.63	48.80	32.13	38.59
BitDelta	32	61.11	12.12	51.83	58.50	25.32	<b>18.30</b>	27.36	11.87	34.38	26.35	32.71
DARE-QT	32	62.17	13.40	57.32	67.70	<b>30.87</b>	17.68	<b>29.76</b>	11.76	42.33	28.24	36.12
Delta-CoMe	32	62.40	12.56	56.71	68.30	27.91	15.52	29.39	10.85	41.40	26.64	35.17
IMPART-QT	32	<b>64.29</b>	<b>13.54</b>	<b>58.54</b>	<b>68.50</b>	<b>30.87</b>	17.65	<b>29.76</b>	<b>12.55</b>	<b>45.84</b>	<b>28.27</b>	<b>36.98</b>

Table 6: Comparison of IMPART-QT and baselines on various tasks across backbones. <sup>†</sup> denotes the uncompressed backbone and fine-tuned models, serving as the reference for quantization. CR<sub>qt</sub> denotes the combined compression ratio with sparsification and quantization. The best results are highlighted in **bold**.

Tasks	Method	16	32	64	128	Avg.
GSM8K	BitDelta	59.89	61.11	61.11	59.14	60.31
	DARE-QT	62.55	62.17	62.09	58.00	61.20
	Delta-CoMe	61.94	62.40	61.62	58.23	61.05
	IMPART-QT	<b>64.22</b>	<b>64.29</b>	<b>62.32</b>	<b>60.35</b>	<b>62.80</b>
	BitDelta	52.44	51.83	51.22	50.00	51.37
HumanEval	DARE-QT	61.59	57.32	56.71	55.49	57.78
	Delta-CoMe	59.15	56.71	52.44	55.49	55.95
	IMPART-QT	<b>62.20</b>	<b>58.54</b>	<b>57.32</b>	<b>56.71</b>	<b>58.69</b>
	BitDelta	25.88	25.32	23.66	22.92	24.45
	DARE-QT	31.79	<b>30.87</b>	28.65	<b>27.73</b>	29.76
IFEval	Delta-CoMe	31.24	27.91	28.10	25.88	28.28
	IMPART-QT	<b>32.16</b>	<b>30.87</b>	<b>30.68</b>	<b>27.73</b>	<b>30.36</b>

Table 7: Performance of IMPART-QT with different compression ratios CR<sub>qt</sub>.

tasks, including GSM8K, HumanEval, and IFEval. We use WizardMath-13B and LLaMA2-Chat-13B as the mathematical and chat-specialized models that are fine-tuned from LLaMA2. Since model merging requires fine-tuned models sharing the same backbone, we fine-tune the LLaMA2-13B backbone on the Magicoder dataset (Wei et al., 2024) to obtain the code specialized model, which we refer to as LlamaCoder. The detailed fine-tuning configuration is shown in the Appendix C.2. We integrate IMPART into two common merging strategies: TA and TIES, and compare IMPART with DARE and no pre-sparsification. Please refer to Appendix C for more details.

**Results** Table 8 summarizes the merging results for IMPART across various tasks and merging strategies. IMPART achieves the highest average scores of 40.98 and 39.99 for TA and TIES, outperforming DARE by 0.46 and 0.78. Compared to no re-sparsification, IMPART improves model merging performance by 0.47 and 1.71 for TA and TIES, respectively. In contrast, DARE shows minimal improvement in TA merging performance. These results underscore the effectiveness of IMPART in improving model merging.

Models	Merge	Mask	GSM8K	HumanEval	IFEval	Avg.
Math	-	No	63.96	-	-	-
Code	-	No	-	52.44	-	-
Chat	-	No	-	-	33.64	-
Chat&Math&Code	TA	No	62.02	30.49	29.02	40.51
		DARE	61.26	<b>31.10</b>	<b>29.21</b>	40.52
		IMPART	<b>63.00</b>	<b>31.10</b>	28.84	<b>40.98</b>
	TIES	No	57.54	24.39	32.90	38.28
		DARE	<b>59.59</b>	24.39	33.64	39.21
		IMPART	58.45	<b>26.22</b>	<b>35.30</b>	<b>39.99</b>

Table 8: Comparison of different sparsification strategies for model merging.

## 8 Related Work

**Model Sparsification** The increasing size of LLMs has made model compression a critical research focus. While traditional model pruning approaches (Li et al., 2018; Lee et al., 2021) remove parameters based on magnitude, they often lead to significant performance degradation when applied to fine-tuned models (Yao et al., 2024). Recent work has instead focused on delta-sparsification, where ERE (Ryu et al., 2023) employs low-rank decomposition of delta weights, and DARE (Yu et al., 2024) demonstrates the effectiveness of random parameter dropping. However, these methods either disregard parameter importance entirely or evaluate it at too coarse a granularity. In contrast, IMPART introduces importance-aware sparsification that assesses and prunes individual singular vectors, achieving superior performance.

**Model Quantization** Parameter quantization has emerged as a prominent compression technique, with GPTQ (Frantar et al., 2023) pioneering error-minimizing low-bit-width approaches. Subsequent innovations have extended to mixed-precision quantization across model weights (Dettmers et al., 2023), activations (Shen et al., 2023), and layers (Bablani et al., 2024). In the context of delta parameters, initial approaches like GPT-Zip (Isik et al., 2023) and DeltaZip (Yao et al., 2024) achieved



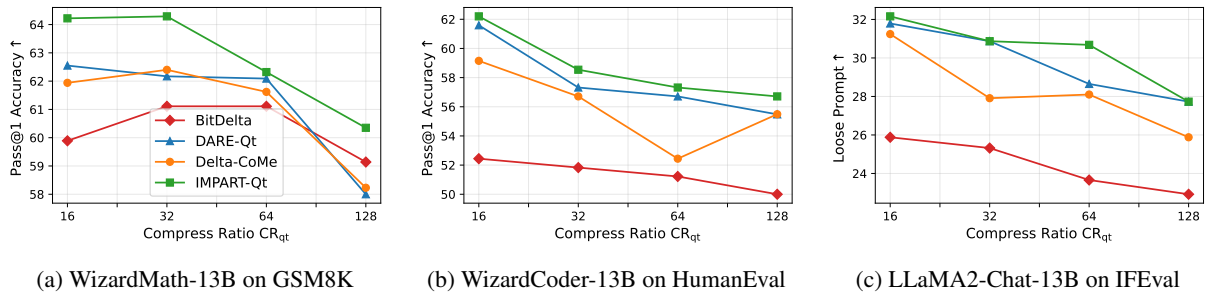


Figure 4: Comparative evaluation of IMPART against state-of-the-art quantization methods across mathematical reasoning, code generation, and chat tasks (more detailed discussions are in Section 7.1).

2-bit compression through GPTQ extensions and structured pruning, while BitDelta (Liu et al., 2024) advanced to 1-bit compression using trainable scaling factors. Delta-CoMe (Ping et al., 2024) further enhanced efficiency by introducing varying bit-width representations for singular vectors. IMPART builds upon these advances by integrating importance-aware sparsification with Delta-CoMe, establishing new SOTA compression performance.

**Model Merging** The proliferation of task-specific models (Luo et al., 2025, 2023; Wei et al., 2024) from open-source pre-trained backbones (Touvron et al., 2023; Grattafiori et al., 2024; Jiang et al., 2023) has motivated efficient model merging techniques to reduce deployment costs. While initial approaches like parameter averaging (Wortsman et al., 2022; Ilharco et al., 2023) demonstrated the potential of combining delta parameters, subsequent methods addressed parameter conflicts through Fisher information matrices (Matena and Raffel, 2022), linear regression (Jin et al., 2023), and magnitude-based parameter selection (Yadav et al., 2023). Although DARE (Yu et al., 2024) introduced random delta weight dropping during merging, it overlooks parameter importance. IMPART advances this direction by incorporating importance-aware sparsification in the SVD space, leading to more effective model merging.

## 9 Conclusion

We introduced IMPART, a novel importance-aware delta-sparsification approach for efficient model compression and merging in large language models. By leveraging singular value decomposition to adaptively determine sparsity ratios based on parameter importance, IMPART effectively preserves critical task-specific knowledge while achieving significant sparsification. Our comprehensive experiments in mathematical reasoning, code genera-

tion, and chat tasks demonstrate that IMPART consistently outperforms existing sparsification methods. Additionally, IMPART can be integrated with state-of-the-art delta-quantization and model merging techniques, achieving new benchmarks in both delta-quantization and model merging.

## Limitations

While we demonstrate the effectiveness of IMPART in compressing and merging LLMs, several limitations remain. First, IMPART treats all weight matrices equally and does not consider the potential benefits of layer-wise pruning, which has been shown to improve compression performance and model efficiency (Li et al., 2024; Dumitru et al., 2024; Wang et al., 2025a; Li et al., 2025). Future work could explore fine-grained, layer-specific sparsification strategies for different layers and matrices. Second, IMPART requires a validation set to determine the optimal hyperparameter tuning. While this is common practice in model compression (Frantar et al., 2023; Ping et al., 2024), it may not always align perfectly with test performance. Nevertheless, IMPART consistently achieves state-of-the-art results across diverse tasks and various hyperparameter configurations, demonstrating robustness. Third, our primary focus is on reducing the storage cost of task-specific delta weights, leaving inference-speed improvements unexplored. Future work could develop method-specific kernels to fully harness the potential gains in inference efficiency.

## Acknowledgements

This project was supported by National Natural Science Foundation of China (No. 62306132) and Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515011564). We thank the anonymous reviewers for their insightful feedback on this work.

## References

- Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, et al. 2024. [Phi-4 technical report](#). *Preprint*, arXiv:2412.08905.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. [Program synthesis with large language models](#). *Preprint*, arXiv:2108.07732.
- Deepika Bablani, Jeffrey L. McKinstry, Steven K. Esser, Rathinakumar Appuswamy, and Dharmendra S. Modha. 2024. [Efficient and effective methods for mixed precision neural network quantization for faster, energy-efficient inference](#). *Preprint*, arXiv:2301.13330.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- 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. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, et al. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, and Han Bao et al. 2024. [Deepseek-v3 technical report](#). *Preprint*, arXiv:2412.19437.
- Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefer, and Dan Alistarh. 2023. [Spqr: A sparse-quantized representation for near-lossless llm weight compression](#). *Preprint*, arXiv:2306.03078.
- Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. 2024a. [Parameter competition balancing for model merging](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 84746–84776. Curran Associates, Inc.
- Mingzhe Du, Anh Tuan Luu, Bin Ji, Qian Liu, and See-Kiong Ng. 2024b. [Mercury: A code efficiency benchmark for code large language models](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. [Length-controlled alpacaeval: A simple way to debias automatic evaluators](#). *Preprint*, arXiv:2404.04475.
- Razvan-Gabriel Dumitru, Paul-Ioan Clotan, Vikas Yadav, Darius Peteleaza, and Mihai Surdeanu. 2024. [Change is the only constant: Dynamic llm slicing based on layer redundancy](#). *Preprint*, arXiv:2411.03513.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2023. [OPTQ: Accurate quantization for generative pre-trained transformers](#). In *The Eleventh International Conference on Learning Representations*.
- Shangqian Gao, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 2024. [Adaptive rank selections for low-rank approximation of language models](#). In *NAACL-HLT*, pages 227–241.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and et al. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. [Editing models with task arithmetic](#). In *The Eleventh International Conference on Learning Representations*.
- Berivan Isik, Hermann Kumbong, Wanyi Ning, Xiaozhe Yao, Sanmi Koyejo, and Ce Zhang. 2023. [GPT-zip: Deep compression of finetuned large language models](#). In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, et al. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. [Follow-Bench: A multi-level fine-grained constraints following benchmark for large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.
- Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. [Dataless knowledge fusion by merging weights of language models](#). In *The Eleventh International Conference on Learning Representations*.

- Kimi Team et al. 2025. [Kimi k1.5: Scaling reinforcement learning with llms](#). *Preprint*, arXiv:2501.12599.
- Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. 2021. [Layer-adaptive sparsity for the magnitude-based pruning](#). In *International Conference on Learning Representations*.
- Guiying Li, Chao Qian, Chunhui Jiang, Xiaofen Lu, and Ke Tang. 2018. [Optimization based layer-wise magnitude-based pruning for dnn compression](#). In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2383–2389. International Joint Conferences on Artificial Intelligence Organization.
- Yixia Li, Boya Xiong, Guanhua Chen, and Yun Chen. 2024. [SeTAR: Out-of-distribution detection with selective low-rank approximation](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Zhiteng Li, Mingyuan Xia, Jingyuan Zhang, Zheng Hui, Linghe Kong, Yulun Zhang, and Xiaokang Yang. 2025. [Adasvd: Adaptive singular value decomposition for large language models](#). *Preprint*, arXiv:2502.01403.
- James Liu, Guangxuan Xiao, Kai Li, Jason D. Lee, Song Han, Tri Dao, and Tianle Cai. 2024. [Bitdelta: Your fine-tune may only be worth one bit](#). *Preprint*, arXiv:2402.10193.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2025. [Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct](#). *Preprint*, arXiv:2308.09583.
- Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. [Wizardcoder: Empowering code large language models with evol-instruct](#). *Preprint*, arXiv:2306.08568.
- Michael Matena and Colin Raffel. 2022. [Merging models with fisher-weighted averaging](#). *Preprint*, arXiv:2111.09832.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.
- Bowen Ping, Shuo Wang, Hanqing Wang, Xu Han, Yuzhuang Xu, Yukun Yan, Yun Chen, Baobao Chang, Zhiyuan Liu, and Maosong Sun. 2024. [Delta-come: Training-free delta-compression with mixed-precision for large language models](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Simo Ryu, Seunghyun Seo, and Jaejun Yoo. 2023. [Efficient storage of fine-tuned models via low-rank approximation of weight residuals](#). *Preprint*, arXiv:2305.18425.
- Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. 2024. [Compressing large language models using low rank and low precision decomposition](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. 2024. [The truth is in there: Improving reasoning in language models with layer-selective rank reduction](#). In *The Twelfth International Conference on Learning Representations*.
- Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. 2023. [Agile-quant: Activation-guided quantization for faster inference of llms on the edge](#). *Preprint*, arXiv:2312.05693.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Boyao Wang, Rui Pan, Shizhe Diao, Xingyuan Pan, Jipeng Zhang, Renjie Pi, and Tong Zhang. 2025a. [Adapt-pruner: Adaptive structural pruning for efficient small language model training](#). *Preprint*, arXiv:2502.03460.
- Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025b. [SVD-LLM: Truncation-aware singular value decomposition for large language model compression](#). In *The Thirteenth International Conference on Learning Representations*.
- Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. [Magicoder: Empowering code generation with oss-instruct](#). *Preprint*, arXiv:2312.02120.
- Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. [Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time](#). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.
- Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. [TIES-merging: Resolving interference when merging models](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu,

Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, et al. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Xiaozhe Yao, Qinghao Hu, and Ana Klimovic. 2024. *Deltazip: Efficient serving of multiple full-model-tuned llms*. *Preprint*, arXiv:2312.05215.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*. PMLR.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. *Instruction-following evaluation for large language models*. *Preprint*, arXiv:2311.07911.

## A More Details for IMPART

### A.1 Sparsity Ratio Allocation

Algorithm 1 shows the details of sparsity ratio allocation across singular vectors for a given target ratio of  $\alpha$ . For simplicity, we only present the case of square matrices.

---

#### Algorithm 1 Sparsity Ratios Computation

---

**Require:** Singular values  $\{\sigma_i\}_{i=1}^n$ , Target sparsity ratio  $\alpha$ , Pre-prune ratio  $\beta$ , Rescale parameter  $C$   
**Ensure:** Sparsity ratio list  $P$  for singular vectors in  $U$  and  $V$   
1:  $\alpha \leftarrow (1 + \alpha)/2$   $\triangleright$  Update sparsity ratio for  $U$  and  $V$   
2: Let  $r = \lfloor n \cdot (1 - \beta) \rfloor$   
3: **for**  $i \leftarrow r + 1$  **to**  $n$  **do**  $p_i \leftarrow 1$   $\triangleright$  Pre-prune  
4:  $\gamma \leftarrow \min\left(\frac{\alpha - \beta}{1 - \beta} \cdot \frac{r}{\sum_{i=1}^r (1 - (\frac{\sigma_i}{\sigma_1})^C)}, \frac{1}{(1 - (\frac{\sigma_r}{\sigma_1})^C)}\right)$   
5: **for**  $i \leftarrow 1$  **to**  $r$  **do**  $p_i \leftarrow (1 - (\frac{\sigma_i}{\sigma_1})^C) \cdot \gamma$   $\triangleright$  Importance-aware sparsification  
6:  $i \leftarrow r$   
7: **while**  $\frac{1}{r} \sum_{k=1}^r p_k < \alpha$  **do**  
8:      $p_i \leftarrow 1$   $\triangleright$  Shift boundary to meet target sparsity ratio  
9:      $i \leftarrow i - 1$   
10: **return**  $P \leftarrow \{p_k\}_{k=1}^n$

---

## B More Details for IMPART-QT

### B.1 GPTQ for Sparse Weight

Delta-CoMe quantizes the left and right singular matrix using GPTQ with a designed mix-precision strategy. However, GPTQ has been primarily confined to dense models. We extend GPTQ to accommodate sparse matrices. Specifically, during the column-by-column quantization process, we apply a sparsification mask to the parameters, ensuring that only those retained after sparsification are subject to quantization. Furthermore, when updating the remaining weights based on quantization error, we compute the error solely on the retained parameters. The detailed algorithm is presented in Algorithm 2.

---

### Algorithm 2 GPTQ for Sparse Weight

---

**Require:** Weight to be quantized  $W$  and its corresponding mask  $M$ , Inverse Hessian  $H^{-1} = (2XX^T + \lambda I)^{-1}$ , and blocksize  $B$   
**Ensure:** Quantized weight  $Q$   
1: Initialize  $Q \leftarrow \mathbf{0}_{d_{\text{row}} \times d_{\text{col}}}$   $\triangleright$  Quantized output  
2: Initialize  $E \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$   $\triangleright$  Block quantization errors  
3:  $H^{-1} \leftarrow \text{Cholesky}(H^{-1})$   $\triangleright$  Hessian inverse information  
4: **for**  $i = 0, B, 2B, \dots$  **do**  
5:     **for**  $j = i, \dots, i + B - 1$  **do**  
6:          $M_{\text{tmp}} \leftarrow M_{:,j}$   
7:          $W_{\text{tmp}} \leftarrow W_{:,j} \odot M_{\text{tmp}}$   $\triangleright$  Set the sparsified weight to zero  
8:          $Q_{:,j} \leftarrow \text{quant}(W_{\text{tmp}}) \odot M_{\text{tmp}}$   
9:          $E_{:,j-i} \leftarrow (W_{\text{tmp}} - Q_{:,j}) / [H^{-1}]_{jj}$   $\triangleright$  Quantization error  
10:          $W_{:,j:(i+B)} \leftarrow E_{:,j-i} \cdot H_{j,j:(i+B)}^{-1}$   $\triangleright$  Update weights in block  
11:          $W_{:, (i+B)} \leftarrow E \cdot H_{i:(i+B), (i+B)}^{-1}$   $\triangleright$  Update all remaining weights

---

### B.2 Compression Ratio Allocation

In line with Delta-CoMe, we employ a triple-precision quantization scheme, assigning 8-bit, 3-bit, and 2-bit precision to distinct singular value groups. The first group consists of 2 elements, the second group includes 32 elements, and the remaining elements form the third group. To achieve the target compression ratio  $\text{CR}_{\text{qt}}$  after quantization, the corresponding sparsity ratio  $\alpha$  is calculated using a binary search process, as described in Algorithm 3. For simplicity, we only present the case of square matrices.

---

#### Algorithm 3 Binary Search to Find Overall Sparsify Ratio for Compression with Quantization

---

**Require:** Singular values  $\{\sigma_i\}_{i=1}^n$ , Compression ratio  $\text{CR}_{\text{qt}}$ , Pre-prune ratio  $\beta$ , Rescale parameter  $C$ , Tolerance  $\text{tol}$   
**Ensure:** Overall sparsify ratio  $\alpha$   
1:  $\text{low} \leftarrow 0$ ,  $\text{high} \leftarrow 1$   $\triangleright$  Set the lower and upper bound  
2: **while**  $\text{high} - \text{low} > \text{tol}$  **do**  
3:      $\text{mid} \leftarrow 0.5 \cdot (\text{low} + \text{high})$   $\triangleright$  Compute the midpoint  
4:      $P \leftarrow \text{Algorithm 1}(\{\sigma_i\}_{i=1}^n, \text{mid}, \beta, C)$   
5:      $\alpha_{\text{qt}} \leftarrow \frac{1}{n} \left( \frac{1}{2} \sum_{i=1}^2 (1 - p_i) + \frac{1}{4} \sum_{i=3}^{34} (1 - p_i) + \frac{1}{8} \sum_{i=35}^r (1 - p_i) \right)$   
6:     **if**  $\frac{1}{1 - \alpha_{\text{qt}}} < 2 \cdot \text{CR}_{\text{qt}}$  **then**  
7:          $\text{low} \leftarrow \text{mid}$   $\triangleright$  Update lower bound  
8:     **else**  
9:          $\text{high} \leftarrow \text{mid}$   $\triangleright$  Update upper bound  
10: **return**  $0.5 \cdot (\text{low} + \text{high})$

---

### B.3 The Storage of Sparsification Mask

Technically, a random sparsification of  $U$  (Equation 4) and  $V$  (Equation 5) would necessitate storing sparsity masks for reconstruction (Equation 6), resulting in additional storage overhead. To ad-

dress this issue, we implement a deterministic seeding strategy: we initialize the random seed for  $\xi_k^i$  (Equation 2) using  $\sigma_k$  when sparsifying  $U$ , and use random seed+1 for  $\eta_k^j$  (Equation 3). This approach maintains the independence of  $\xi_k^i$  and  $\eta_k^j$  while enabling the reconstruction of sparsity masks directly from the singular value  $\sigma_k$ , thus avoiding additional storage.

#### B.4 Baselines for IMPART-QT

**BitDelta** BitDelta (Liu et al., 2024) achieves a 1/16 compression ratio by compressing the task vector into  $\mu \odot \text{Sign}(\Delta)$ , where  $\text{Sign}(\cdot)$  denotes the 1-bit element-wise sign of each parameter and  $\mu$  is a trainable scaling factor. In this paper, we further combine BitDelta with DARE to achieve an even higher compression ratio.

**DARE-QT** DARE-QT is the baseline that integrates DARE into GPTQ. DARE first sparsifies the delta parameters, and then GPTQ further quantizes the sparsified delta parameters. To quantize the sparse delta parameters, we use the same version of GPTQ as shown in Appendix B.1. For each compression ratio  $\text{CR}_{\text{qt}}$ , we use GPTQ to quantize the 16-bit parameters into 2/4/8-bit, with the sparsity ratio  $\alpha$  of DARE determined by the target compression ratio  $\text{CR}_{\text{qt}}$ . Then we report the configuration that achieved the best performance on the validation set for each compression ratio  $\text{CR}_{\text{qt}}$ .

**Delta-CoMe** We faithfully implement Delta-CoMe as described in the original paper (Ping et al., 2024), achieving the target compression ratio by adjusting the number of 2-bit singular vectors.

#### B.5 Performance of IMPART-QT Across Compression Ratios

Figure 4 visualizes the performance of IMPART-QT and baselines on different tasks across compression ratios of 16 to 128.

### C More Details for Model Merging

#### C.1 Common Model Merging Methods

**TA** Task Arithmetic (Ilharco et al., 2023) leverages a scaling term to regulate the contributions of the pre-trained backbone and the aggregated delta parameters set, formed by summing  $n$  multiple individual delta parameters:

$$W^{\text{merge}} = W^{\text{base}} + \lambda * \sum_{t=1}^n \Delta W^t, \quad (10)$$

**TIES** TIES-Merging (Yadav et al., 2023) aims to address parameter conflicts in model merging. Given a delta parameter set, it first trims parameters with lower magnitudes,

$$\Delta W^t = \text{trim}(\Delta W^t). \quad (11)$$

Then, TIES elects the sign with the highest total magnitude to resolve sign disagreements:

$$\gamma^t = \text{sgn}(\Delta W^t), \quad (12)$$

$$\gamma^m = \text{sgn}\left(\sum_{t=1}^n \Delta W^t\right). \quad (13)$$

Finally, Parameters with consistent signs are disjointly merged:

$$\mathcal{A} = \{t \in [n] \mid \gamma^t = \gamma^{\text{merge}}\}, \quad (14)$$

$$W^{\text{merge}} = W^{\text{base}} + \lambda * \frac{1}{|\mathcal{A}|} \sum_{t \in \mathcal{A}} \Delta W^t. \quad (15)$$

#### C.2 Details of LlamaCoder

We implement LlamaCoder by full fine-tuning from the Llama2-13B base model using the Magi-coder dataset (Wei et al., 2024). The training process involved 3 epochs, with a batch size of 8, a peak learning rate of 2e-5, and a maximum sequence length of 4096. Note that we do not use WizardCoder-13B as its backbone is Codellama-13B.

#### C.3 Hyperparameter Selection

We follow DARE (Yu et al., 2024; Du et al., 2024a) for hyperparameter search. Specifically, we perform a grid search to optimize the hyperparameters of TA and TIES. Specifically, for both methods, the scaling term is selected from the set {0.4, 0.6, 0.8, 1.0, 1.2}, and for TIES, the retain ratio of the largest-magnitude parameters is chosen from {0.4, 0.6, 0.8}. When incorporating the sparsification methods DARE and IMPART into TA/TIES, we use the pre-selected hyperparameters of TA/TIES and search for the optimal sparsification ratios from {0.1, 0.3, 0.5, 0.7, 0.9} to save computation.