MaZO: Masked Zeroth-Order Optimization for Multi-Task Fine-Tuning of Large Language Models

Zhen Zhang[1], **Yifan Yang**[1], **Kai Zhen**[2], **Nathan Susanj**[2], **Athanasios Mouchtaris**[2], **Siegfried Kunzmann**[2], **Zheng Zhang**[1]
[1]University of California, Santa Barbara
[2]Amazon AGI
zhen_zhang@ucsb.edu, zhengzhang@ece.ucsb.edu

Abstract

Large language models have demonstrated exceptional capabilities across diverse tasks, but their fine-tuning demands significant memory, posing challenges for resource-constrained environments. Zeroth-order (ZO) optimization provides a memory-efficient alternative by eliminating the need for backpropagation. However, ZO optimization suffers from high gradient variance, and prior research has largely focused on single-task learning, leaving its application to multi-task learning unexplored. Multi-task learning is crucial for leveraging shared knowledge across tasks to improve generalization, yet it introduces unique challenges under ZO settings, such as amplified gradient variance and collinearity. In this paper, we present MaZO, the first framework specifically designed for multi-task LLM fine-tuning under ZO optimization. MaZO tackles these challenges at the parameter level through two key innovations: a weight importance metric to identify critical parameters and a multi-task weight update mask to selectively update these parameters, reducing the dimensionality of the parameter space and mitigating task conflicts. Experiments demonstrate that MaZO achieves state-of-the-art performance, surpassing even multi-task learning methods designed for first-order optimization.

1 Introduction

Large language models (LLMs) have revolutionized natural language processing, enabling breakthroughs in various applications (Anthropic, 2024; DeepMind, 2024; OpenAI, 2024; Bai et al., 2023). However, the large sizes of LLMs pose significant memory challenges during training. Traditional first-order (FO) optimization uses backpropagation, which requires substantial memory to store intermediate activations and gradients (Rostam et al., 2024; Kundu et al., 2024). This issue is especially pronounced in fine-tuning tasks on resource-constrained platforms (e.g. low-end GPUs or edge devices) (Zhuang et al., 2024). Moreover, certain hardware platforms lack software support (e.g. automatic differentiation) for backpropagation (Bergholm et al., 2018), further restricting FO methods. Although parameter-efficient fine-tuning methods have alleviated some of these challenges, they still require multiple times the memory of inference (Bai et al., 2024a; Zhang et al., 2024b).

Zeroth-order (ZO) optimization provides a memory-efficient alternative by estimating gradients via forward passes only. Recent advances, such as MeZO (Malladi et al., 2023), have reduced memory usage to inference levels while achieving strong performance in LLM fine-tuning. However, the gradient variance in ZO methods is proportional to the number of perturbed parameters, which makes ZO methods struggle with high-dimensional parameter spaces, leading to slower convergence, increased gradient estimation variance, and hard to scale up (Chen et al., 2024b). Although recent work (Liu et al., 2024b; Yang et al., 2024c; Chen et al., 2023; Liu et al., 2024b; Yu et al., 2024) has addressed some of these issues, most ZO methods focus on single-task learning, leaving their application to multi-task learning largely unexplored.

Multi-task learning is a key paradigm in LLMs to enable shared representations across diverse downstream tasks. This approach improves generalization, reduces the need for task-specific models, and improves performance in a wide range of applications (Zhang et al., 2023; Radford et al., 2019). Despite its advantages, multi-task learning also introduces inherent challenges, particularly when tasks exhibit conflicting objectives. These conflicts arise when the optimization signals from different tasks are misaligned, leading to competing gradients that prevent the model from learning effectively across all tasks (Sener and Koltun, 2018; Mahapatra and Rajan, 2020; Crawshaw, 2020; Zhou et al., 2022; Shi et al., 2023).

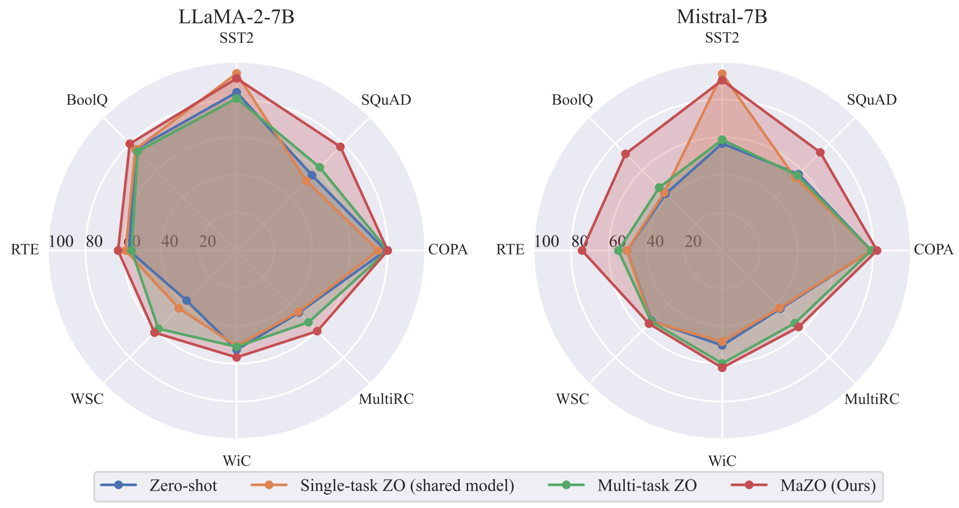

Figure 1: Radar chart comparing the performance of our MaZO method with other methods on LLaMA-2-7B and Mistral-7B. Larger is better. Shared model means we train the model on one task and test it on all tasks.

The issue of conflicting gradients is further exacerbated in scenarios involving ZO optimization (Liu et al., 2020; Malladi et al., 2023). The high gradient variance in ZO methods can amplify inter-task conflicts and make it even more difficult to balance competing objectives (Zhang et al., 2024a). Furthermore, ZO methods suffer from collinearity in gradient estimates (see Section 2.2), where aggregated gradient directions lack diversity, and higher rank in Hessian matrix (see Section 3.1), where slower decay of eigenvalues in multi-task learning makes the convergence slow. A primary experiment demonstrated in Figure 1 shows that vanilla multi-task ZO optimization is only slightly better than zero-shot on average and is even worse on many tasks.

To address these challenges, we propose Masked Zeroth-Order Optimization (MaZO), a novel framework designed for multi-task fine-tuning under ZO settings. MaZO tackles the problem at parameter level, which introduces two key innovations: (1) a weight importance metric that identifies critical parameters for each task, and (2) a multi-task weight update mask that selectively updates these parameters while freezing others. By focusing on the most important parameters, MaZO reduces the dimension of parameter space, mitigating the high variance of ZO fine-tuning while preserving the model capacity. Moreover, unlike traditional approaches dynamic weighting (Chen et al., 2018; Liu et al., 2024a; Aghajanzadeh et al., 2023), which are trivial in ZO settings because of collinearity, MaZO balances multi-task learning conflicts from the perspective of weight. It activates distinct parameter subsets for different tasks based on their importance scores, allowing MaZO to allocate more capacity to tasks that require more updates.

Paper Contributions. This paper makes the following novel contributions:

- **First ZO-based multi-task fine-tuning framework**: We propose Masked Zeroth-Order Optimization (MaZO), the first framework specifically designed for multi-task LLM fine-tuning under ZO optimization.
- **Task conflict resolution at the parameter level**: MaZO addresses inter-task conflicts by selectively activating critical parameters for each task. This parameter-level approach ensures balanced optimization across tasks under ZO settings.
- **State-of-the-art performance**: Comprehensive experiments on LLaMA-2-7B and Mistral-7B demonstrate that MaZO achieves state-of-the-art results in multi-task fine-tuning under ZO settings, outperforming multi-task learning methods designed for first-order (FO) optimization.

2 Preliminaries and Related Work

2.1 Zeroth-Order Optimization

Zeroth-order (ZO) optimization estimates gradients using forward passes only. A common approach for ZO gradient estimation is the simultaneous perturbation stochastic approximation (Spall, 1992), which serves as a randomized gradient estimator. Consider a model with parameters $\theta \in \mathbb{R}^d$ and a loss function $\mathcal{L}(\theta)$. Using Taylor expansion, the randomized gradient can be estimated by perturbing θ with random noise $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I}_d)$ and computing forward and reverse losses:

$$\widehat{\nabla}\mathcal{L}(\theta) = \frac{\mathcal{L}(\theta + \epsilon\mathbf{z}) - \mathcal{L}(\theta - \epsilon\mathbf{z})}{2\epsilon}\mathbf{z}, \quad (1)$$

where ϵ is a small scalar. The expectation of $\widehat{\nabla}\mathcal{L}(\theta)$ matches the smoothed version of the true gradient. During training, zeroth-order stochastic gradient descent (ZO-SGD) updates parameters as:

$$\theta = \theta - \eta\widehat{\nabla}\mathcal{L}(\theta), \quad (2)$$

where η is the learning rate.

Recent advances have improved ZO optimization for large-scale applications. For example, MeZO (Malladi et al., 2023) reduces memory usage by regenerating random perturbations $\mathbf{z}$ using random seeds instead of storing them. ZO optimization offers significant advantages for fine-tuning LLMs, as it avoids memory-intensive backpropagation (Liu et al., 2020; Zhang et al., 2024b). Despite these advantages, the gradient variance of ZO optimization increases linearly with the dimensionality of the parameter space. This leads to slower convergence and difficulties in large-scale training (Chen et al., 2024b). To address these challenges, various methods have been proposed. These include the design of advanced ZO optimizers (Zhao et al., 2024; Jiang et al., 2024; Chen et al., 2019); dimensionality reduction techniques (Liu et al., 2024b; Wang et al., 2024; Yang et al., 2024c; Guo et al., 2024); hybrid approaches like Addax (Li et al., 2024); full-batch gradient estimation (Gautam et al., 2024); exploiting low-rank structures (Zhao et al., 2023; Yu et al., 2024), and using orthogonal random directions (Kozak et al., 2023).

While these methods have advanced ZO in various ways, they do not specifically address the unique challenges of multi-task learning.

2.2 Multi-task Learning

Multi-task learning aims to improve generalization performance by jointly learning T related tasks through shared parameters (Chen et al., 2024a). Classical multi-task learning minimizes a weighted combination of task-specific losses:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} w_t\mathcal{L}^t(\theta), \text{s.t.} \sum_{t=1}^{T} w_t = 1,\ w_t \geq 0, \quad (3)$$

where $\mathcal{L}^t(\theta)$ represents the learning loss for a single task t. Parameter updates are performed using gradient descent.

Multi-task learning under FO optimization has been widely studied, with different technical routes: (1) dynamic weight, which adjusts the weight of different tasks by gradients (Chen et al., 2018; Sener and Koltun, 2018; Mao et al., 2022), loss (Liu et al., 2019, 2024a; Kongyoung et al., 2020; Gong et al., 2024) or uncertainty (Aghajanzadeh et al., 2023); (2) gradient manipulation (Désidéri, 2012; Liu et al., 2021; Yu et al., 2020); (3) data mixing and scheduling (Bai et al., 2024b; Ahmadian et al., 2024); (4) learning shared and specific knowledge with model architecture based on LoRA (Feng et al., 2024; Yang et al., 2024b; Wang et al., 2023) or MoE (Liu et al., 2023; Gupta et al., 2022); (5) model merging (Yang et al., 2023).

3 The MaZO Framework

3.1 Challenges in ZO Multi-Task Fine Tuning

Under ZO optimization, multi-task learning faces unique challenges. Specifically, task-specific ZO gradient estimates exhibit fundamental collinearity, as the aggregated multi-task learning gradient aligns with the shared random perturbation $\mathbf{z}$:

$$\begin{aligned}\mathbf{g} &= \sum_{t=1}^{T} w_t\mathbf{g}^t \\ &= \left(\sum_{t=1}^{T} w_t \frac{\mathcal{L}^t(\theta + \epsilon\mathbf{z}) - \mathcal{L}^t(\theta - \epsilon\mathbf{z})}{2\epsilon}\right)\mathbf{z}. \quad (4)\end{aligned}$$

Here $\mathbf{g}$ and $\mathbf{g}^t$ are gradients of multi-task learning and of task t, respectively. This collinearity results in a lack of directional diversity, limiting optimization efficacy. Further discussion can be found in Appendix H.

As explained in (Malladi et al., 2023), the surprising success of ZO optimization in LLM fine-tuning is due to the low-rank property of the Hessian matrix. Based on (3), the Hessian matrix in multi-task fine-tuning can be written as

$$\mathbf{H} = \sum_{t=1}^{T} w_t\mathbf{H}^t, \quad (5)$$

where $\mathbf{H}^t$ is the Hessian associated with single-task learning loss $\mathcal{L}^t$. Although $\mathbf{H}^t$ has a low rank in the fine-tuning process, $\mathbf{H}$ can have a much higher rank due to the weighted sum of T task-specific Hessian matrices. Figure 3 empirically verifies our theoretical claim: the Hessian in multi-task learning exhibits a broader eigenvalue spectrum than single-task learning, leading to a higher effective rank. This further slows down the convergence of ZO in multi-task LLM fine-tuning.

To address the above challenges, we propose **Masked Zeroth-Order Optimization (MaZO)**.

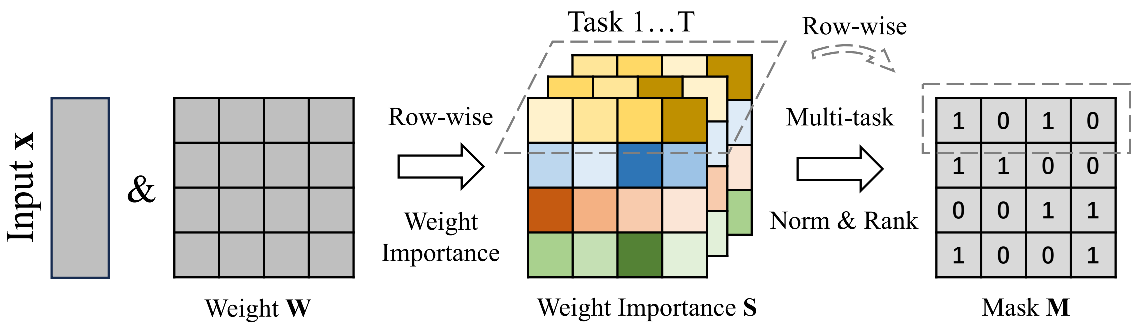

Figure 2: Diagram of our MaZO method. The weight importance scoring and weight update mask is calculated row-wise. The weight importance for each task is calculated independently, and only from the input and weight.

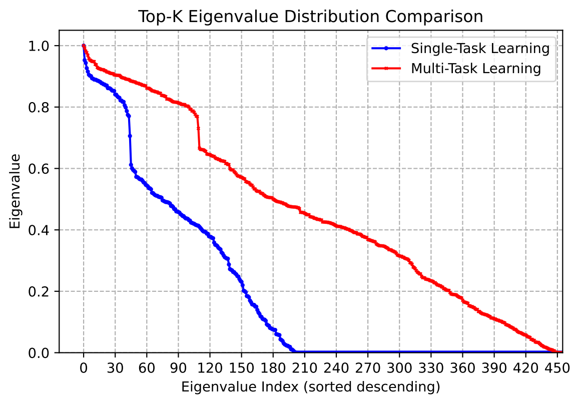

Figure 3: Top-K eigenvalue distribution of the Hessian matrices in multi-task learning and single-task learning. These eigenvalue are normalized by dividing by the maximum value. The slower decay of eigenvalues in multi-task learning suggests a higher effective rank, which contributes to the slower convergence of ZO fine-tuning in multi-task scenarios.

Our approach introduces a novel framework that solves multi-task learning at parameter level. MaZO combines weight importance metrics and a multi-task weight update mask. The weight importance is derived using two complementary metrics: (1) a global score which evaluates the theoretical minimum loss when freezing a parameter, (2) a greedy score which quantifies the immediate loss change during a single optimization step. Using these scores, we construct a weight update mask that identifies a subset of critical parameters, enabling effective optimization by reducing dimensionality and variance while balancing the performance among potentially conflicting tasks.

3.2 Multi-Task Weight Update Mask

We first introduce the multi-task weight update mask, assuming the weight importance scores are precomputed. We defer the computation of weight importance scores to the next subsection. In ZO optimization, the variance of an estimated gradient increases with the number of training parameters. Therefore, it is crucial to identify and focus on critical parameters for effective optimization while freezing others (Liu et al., 2024b; Guo et al., 2024).

Suppose that we have a weight importance score matrix $\mathbf{S}^t$ for each task t and a sparsity level ρ. We unfreeze the top $k = \lceil (1-\rho) \cdot N \rceil$ parameters in each row, where N is the total number of parameters in that row. The importance scores are compared row-wise due to the approximations involved in gradient and Hessian estimation following Sun et al. (2023), which will be detailed in Section 3.4.

Since importance scores across tasks are not directly comparable due to differing scales, we normalize the scores row-wise for each task:

$$\hat{\mathbf{S}}_{ij}^t = \frac{\mathbf{S}_{ij}^t - \min(\mathbf{S}_i^t)}{\max(\mathbf{S}_i^t) - \min(\mathbf{S}_i^t)}, \tag{6}$$

where $\mathbf{S}_i^t$ denotes the i-th row of $\mathbf{S}^t$; $\hat{\mathbf{S}}_{ij}^t$ is the normalized score for parameter j in row i for task t. The overall score across tasks is computed as:

$$\mathbf{S} = \sum_{t=1}^{T} \hat{\mathbf{S}}^t. \tag{7}$$

We select the top k parameters based on $\mathbf{S}$ in *each row* to fine-tune, while freezing the others. This selection is represented by a binary mask matrix $\mathbf{M}$, where $\mathbf{M}_{ij} = 1$ indicates that parameter j in row i is unfrozen. The final parameter update is computed as:

$$\mathbf{\Delta W}_{\text{masked}} = \mathbf{\Delta W} \odot \mathbf{M}, \tag{8}$$

where $\odot$ denotes element-wise multiplication. When applied to LoRA (Hu et al., 2021), this becomes:

$$\mathbf{\Delta W}_{\text{masked}} = (\mathbf{A} \cdot \mathbf{B}) \odot \mathbf{M}, \tag{9}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the decomposed matrices of LoRA.

3.3 Weight Importance

The overall importance score for task t combines the normalized global and greedy scores with a weight regularization term:

$$\mathbf{S}^t = \mathbf{S}^t_{\text{global}} + \alpha \mathbf{S}^t_{\text{greedy}} + \beta |\mathbf{W}|, \tag{10}$$

where α and β are hyperparameters controlling the contributions of each component and $|\mathbf{W}|$ is the absolute value of weight. We now describe the computation of two complementary metrics: the **global score** and the **greedy score**.

3.3.1 Global Score

The global score is inspired by the Optimal Brain Surgeon method (Frantar and Alistarh, 2023; Sun et al., 2023; Das et al., 2023). Unlike pruning, which sets the parameters to zero, our approach freezes certain parameters while updating others via perturbation. Consider the Taylor expansion of the loss function of task t:

$$\delta\mathcal{L}^t = (\mathbf{g}^t)^\top \cdot \delta\theta + \frac{1}{2}\delta\theta^\top \cdot \mathbf{H}^t \cdot \delta\theta + \mathcal{O}(\|\delta\theta\|^3),$$

where $\mathbf{H}^t$ is the Hessian matrix of task t and $\mathbf{g}^t = \frac{\partial \mathcal{L}^t_{\text{STL}}}{\partial\theta}$. Freezing a parameter at position m imposes the constraint $\mathbf{I}_m^\top \delta\theta = 0$, where $\mathbf{I}_m$ is an indicator function. The optimization problem becomes:

$$\min_m \left\{ \min_{\delta\theta} \left(\left(\mathbf{g}^t\right)^\top \cdot \delta\theta + \frac{1}{2}\delta\theta^\top \cdot \mathbf{H}^t \cdot \delta\theta \right) \middle| \mathbf{I}_m^\top \cdot \delta\mathbf{w} = 0 \right\}. \tag{11}$$

This formulation seeks to find the parameter position m that, when frozen, results in the maximal decrease in the loss function while allowing other parameters to adjust optimally. The inner optimization determines the best possible parameter updates given the constraint, while the outer optimization identifies the least impactful parameter to fix.

Using Lagrange multipliers, the optimal loss change (global score) is derived as:

$$(\mathbf{S}^t_{\text{global}})_m = \delta\mathcal{L}^t_m = \frac{\left(\mathbf{I}_m^\top \cdot \left(\mathbf{H}^t\right)^{-1} \cdot \mathbf{g}^t\right)^2}{2\left(\left(\mathbf{H}^t\right)^{-1}\right)_{mm}}, \tag{12}$$

This expression quantifies the theoretical maximum decrease in loss when parameter m is fixed, providing a measure of its importance to the overall optimization process. Smaller values indicate less important parameters, which should be frozen.

3.3.2 Greedy Score

Although the global score provides a theoretical measure of parameter importance, it may not suffice because the model may not converge to the optimal situation due to the large variance in the ZO gradient. Therefore, we also introduce a greedy score as a practical complement, which considers the immediate impact of freezing a parameter in a *single optimization step*.

For a gradient descent update with learning rate η and random direction $\mathbf{z}$, the parameter update of task t is approximated as:

$$\delta\theta \approx -\eta \mathbf{z}\mathbf{z}^T \mathbf{g}^t. \tag{13}$$

Substituting $\delta\theta$ and taking the expectation over random directions z, we obtain the expected change in loss:

$$\begin{aligned}\mathbb{E}(\delta\mathcal{L}^t) &= -(\mathbf{g}^t)^T \mathbf{g}^t \cdot \eta \\ &+ \left(\sum_{i=0}^{M} (\mathbf{g}^t_i)^2 \mathbf{H}^t_{ii} + 2(\mathbf{g}^t_i)^T \mathbf{H}^t \mathbf{g}^t\right)\eta^2\end{aligned}$$

where M is the number of parameters in a LLM.

When we freeze a parameter at position m, the change of loss (greedy score) will increase by:

$$\begin{aligned}(\mathbf{S}^t_{\text{greedy}})_m &= \delta\mathcal{L}^t_m \\ &= (\mathbf{g}^t_m)^2\eta + \mathbf{H}^t_{mm}(\mathbf{g}^t_m)^2\eta^2 \\ &\quad - 4\sum_{j=0}^{M} \mathbf{H}^t_{mj}(\mathbf{g}^t_m)(\mathbf{g}^t_j)\eta^2\end{aligned} \tag{14}$$

Parameters with lower $\mathbf{S}^t_{\text{greedy}}$ values are considered less important for the current optimization step and are better candidates for freezing during multi-task learning.

3.4 Implementation

To avoid the huge cost of computing the full gradient and Hessian, we adopt a row-wise approximation strategy. For a linear layer $\mathbf{y} = \mathbf{W}\mathbf{x}$, focusing on a single row $\mathbf{w}_i$, the output is $\mathbf{y}_i = \mathbf{w}_i\mathbf{x}$. Performing a Taylor expansion of the loss $\mathcal{L}$ with respect to y_i, we find that both the first-order gradient $\nabla\mathcal{L}(\mathbf{y}_i)$ and second-order derivative $\nabla^2\mathcal{L}(\mathbf{y}_i)$ are scalars. Substituting $\Delta\mathbf{y}_i = \Delta\mathbf{w}_i\mathbf{x}$, the gradient and Hessian with respect to $\mathbf{w}_i$ are:

$$\mathbf{g}^t = \frac{\partial\mathcal{L}}{\partial\mathbf{w}_i} = \nabla\mathcal{L}(\mathbf{y}_i)\mathbf{x}, \tag{15}$$

$$\mathbf{H}^t = \frac{\partial^2\mathcal{L}}{\partial\mathbf{w}_i^2} = \nabla^2\mathcal{L}(\mathbf{y}_i)(\mathbf{x}\mathbf{x}^\top). \tag{16}$$

Here, we replace the gradient with $\mathbf{x}$, and the Hessian with $\mathbf{x}\mathbf{x}^\top$ since we only care about the relative value *in a row*. This row-wise approximation significantly reduces computational cost, while still capturing the relative importance of parameters within each row. However, it also restricts the weight-importance comparison to the row direction.

Overall Algorithm Flow. The pseudo-code of the whole MaZO fine-tuning framework is shown as Algorithm 1 in Appendix C.

4 Experiments

4.1 Experimental Setup

We perform multi-task fine-tuning on three widely used decoder-only pretrained language models: LLaMA-2-7B (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023) and Qwen2.5-32B (Yang et al., 2024a).

Tasks. We evaluate our approach on a diverse set of natural language understanding (NLU) and natural language generation (NLG) tasks from the GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) benchmarks. Specifically, for NLU, we include SST-2, BoolQ, RTE, WSC, WiC, MultiRC, and COPA, covering various classification and reasoning tasks. For NLG, we use SQuAD for question answering. Additionally, we evaluate Qwen2.5-32B on the MMLU dataset, which consists of 57 tasks and 14k test examples, showcasing the scalability of our method. Details on datasets and evaluation metrics are in Appendix G.

Reproducibility. Comprehensive details essential for reproducing our results, along with the ablation study, are presented in Appendix B.

Baselines. We compare MaZO with several baselines. First, we include vanilla ZO optimization combined with traditional multi-task learning (MTL-ZO) techniques as a direct comparison to MaZO in the ZO setting. Second, we evaluate single-task learning (STL-ZO), where models are trained individually on each task to provide an *upper bound* for task-specific performance without multi-task conflicts, as well as a single-task transfer baseline, where the model is trained on a single task (SST-2) using vanilla ZO optimization and evaluated across all tasks to highlight the limitations of single-task training in multi-task scenarios. Third, we include LoRA fine-tuning (Hu et al., 2021), a parameter-efficient approach, and extend MaZO to update LoRA matrices under ZO settings.

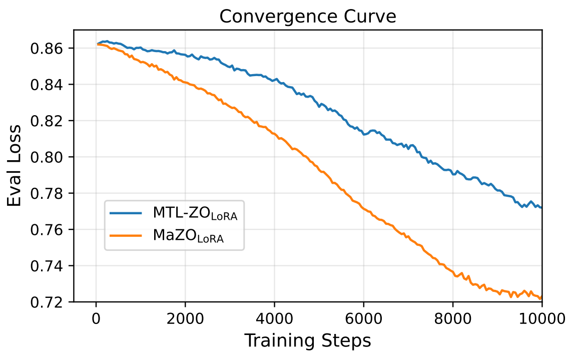

Figure 4: The convergence curve of (1) vanilla multi-task ZO fine-tuning with LoRA, (2) MaZO with LoRA.

Finally, we compare MaZO against state-of-the-art first-order (FO) multi-task learning methods, including CoBa (Gong et al., 2024), FAMO (Liu et al., 2024a), and MTL-LoRA (Yang et al., 2024b), to its compatibility with ZO optimization. Details are discussed in Appendix D. These baselines provide a comprehensive comparison for assessing MaZO's effectiveness and robustness in addressing the challenges of ZO-based multi-task learning.

4.2 Results on LLaMA-2-7B

MaZO Outperforms Competitors. The results for LLaMA-2-7B are presented in Table 1. Vanilla multi-task ZO optimization shows only slight improvements over the zero-shot baseline, highlighting its inability to effectively address multi-task conflicts under ZO settings. Similarly, vanilla single-task ZO optimization with a shared model fails to generalize effectively across multiple tasks, underscoring the inherent challenges of ZO optimization in multi-task scenarios. In contrast, our proposed MaZO framework achieves the **highest average performance** across all tasks and demonstrates a **balanced** performance profile. These results validate MaZO's ability to mitigate inter-task conflicts and optimize multi-task learning by selectively focusing on critical parameters. The effectiveness of MaZO is further evident in its superior performance in both full-model ZO fine-tuning and LoRA-based fine-tuning, with particularly pronounced gains in the latter. This underscores MaZO's flexibility and its compatibility with parameter-efficient fine-tuning techniques.

Dimensionality Reduction Enhances Multi-Task Learning. The application of LoRA to ZO fine-tuning significantly improves the performance of multi-task learning. This improvement can be attributed to LoRA's ability to reduce the dimensionality of the parameter space, thereby lowering

Task **Task Type**	**SST-2**	**BoolQ**	**RTE**	**WSC**	**WiC**	**MultiRC**	**COPA** *– Multiple Choice –*	**SQuAD** *– Generation –*	**Avg**
	Classification								
STL-ZO (one model per task)	93.8	83.0	73.5	51.3	62.1	61.0	86.0	79.6	73.8
Zero-Shot	83.8	75.8	57.0	37.5	52.6	46.6	79.0	56.4	61.1
ICL	93.7	78.7	61.2	47.2	59.9	54.3	80	57.7	66.6
STL-ZO(shared model)	93.8	75.8	58.8	43.3	50.6	46.0	75.0	52.2	61.9
MTL-ZO	80.6	74.2	55.6	58.6	51.0	53.8	**80.0**	62.3	64.5
MTL-ZO$_{\text{LoRA}}$	90.2	**80.0**	61.0	54.8	56.6	58.0	76.0	74.8	68.9
MTL-ZO$_{\text{MTL-LoRA}}$	88.4	76.8	60.2	60.6	**57.6**	**61.6**	79.0	59.4	68.0
MTL-ZO$_{\text{CoBa}}$	82.8	75.3	56.8	60.6	53.4	55.8	77.0	57.6	64.9
MTL-ZO$_{\text{FAMO}}$	91.0	77.6	59.4	56.5	53.4	50.6	78.0	53.9	65.1
MaZO	90.6	76	**62.8**	56.7	52.6	58.6	**82.0**	55.5	66.9
MaZO$_{\text{LoRA}}$	**91.2**	**80.0**	**62.8**	**61.5**	56.6	60.4	80.0	**77.7**	**71.3**

Table 1: Performance comparison across tasks using different methods on LLaMA-2-7B. The average score (Avg) is computed across all tasks. Metrics for these tasks are consistent with MeZO (Malladi et al., 2023). *Shared model* indicates training on a single task (SST-2) and testing on all tasks. *ICL* refers to in-context learning. *STL* represents single-task learning and *MTL* represents multi-task learning.

the variance of gradient estimates. These findings reinforce the validity of MaZO's masking strategy, which optimizes multi-task learning by focusing on a reduced set of critical parameters.

FO multi-task learning methods do not apply to ZO. Multi-task learning methods originally developed for first-order (FO) optimization, such as CoBa and FAMO, do not achieve effective performance in the ZO setting. This can be attributed to their inability to resolve multi-task conflicts due to the collinearity problem in ZO gradient estimates. Under the ZO framework, FO methods can only adjust the magnitude of the approximated gradient, but not its direction, resulting in performance degradation. Additionally, MTL-LoRA, the multi-task version of LoRA does not significantly enhance performance in the ZO setting. This may be due to the sensitivity of task-specific weights and the diagonal transformation matrix to noise. Perturbation-based optimization, as used in ZO, introduces excessive variance, which undermines the effectiveness of these FO-based methods.

4.3 Results on Mistral-7B

The results for Mistral-7B in Table 2 reveal trends similar to those observed with LLaMA-2-7B. Despite the relatively low zero-shot performance of Mistral-7B, vanilla multi-task learning ZO fails to deliver substantial improvements. This underscores the inherent challenges of ZO-based multi-task learning. In contrast, MaZO consistently outperforms all other methods. Its ability to mitigate ZO-specific challenges is evident in its superior performance, further validating MaZO as a state-of-the-art solution for ZO-based multi-task learning.

4.4 Results on Qwen2.5-32B

The results of MMLU benchmark on Qwen2.5-32B are demonstrated in Table 3 that directly applying multi-task learning (e.g., MTL-ZO) can lead to a performance drop compared to the zero-shot baseline. In contrast, our MaZO-based methods consistently improve performance across settings, surpassing both the zero-shot and standard multi-task learning approaches.

4.5 Computational Performance

Figure 4 shows that MaZO converges faster and achieves a significantly lower loss compared to traditional multi-task ZO fine-tuning methods. This holds true both with and without LoRA. This improvement can be attributed to the mask mechanism in MaZO, which focuses on optimizing the most critical parameters, thereby reducing gradient noise, balancing the inter-task conflicts, and accelerating convergence.

To evaluate the efficiency of MaZO, we compare its memory usage, search time, and training time against baseline vanilla multi-task learning ZO methods, both with and without LoRA. Table 4 summarizes the results. The search time introduced by MaZO is negligible compared to the overall training time. MaZO incurs a slight increase in memory usage (approximately 10%) compared to baseline multi-task learning ZO methods. This is primarily due to the additional storage required for the weight update mask. However, this increase is marginal and does not significantly impact the overall memory efficiency, especially when combined with LoRA, where the parameter space is already

Task	**SST-2**	**BoolQ**	**RTE**	**WSC**	**WiC**	**MultiRC**	**COPA**	**SQuAD**	**Avg**
Task Type	*Classification*						*– Multiple Choice –*	*– Generation –*	
STL-ZO (one model per task)	93.6	77.8	74.2	55.3	62.1	62.7	88.0	76.5	73.8
Zero-Shot	56.7	42.4	50.5	52.8	50.3	43.6	79.0	57.2	54.1
ICL	62.3	46.1	56.0	53.2	61.4	53.4	79.0	62.3	59.2
MTL-ZO	58.7	47.2	55.0	53.2	59.8	54.4	79.0	56.3	58.0
MTL-ZO$_{\text{LoRA}}$	89.3	**73.2**	71.5	51.3	58.1	53.4	80.0	**73.5**	68.7
MaZO	83.4	56.3	60.2	54.8	58.1	55.8	79	59.4	63.4
MaZO$_{\text{LoRA}}$	**90.2**	72.4	**74.2**	**54.8**	**62.1**	**57.3**	**82.0**	**73.5**	**70.8**

Table 2: Performance comparison across tasks using different methods on Mistral-7B. The setting and notation are the same as Table 1. We exclude the FO MTL methods as they do not have significant improvement.

Method	MMLU Score
Zero-Shot	83.1
MTL-ZO	81.2
MTL-ZO$_{\text{LoRA}}$	83.4
MTL-ZO$_{\text{FAMO}}$	82.7
MTL-ZO	83.5
MTL-ZO$_{\text{LoRA}}$	**84.1**

Table 3: MMLU Scores of different methods on Qwen2.5-32B.

Method	Memory (GB)	Search Time (min)	Training Time (h)
MTL-ZO	29.0	-	14.3
MaZO	33.3	42	16.6
MTL-ZO$_{\text{LoRA}}$	31.2	-	13.7
MaZO$_{\text{LoRA}}$	33.9	8.5	14.1

Table 4: Comparison of memory usage, search time, and training time between MTL-ZO and MaZO, with and without LoRA. While MaZO introduces marginal memory and runtime overhead due to the mask storage and search process, it achieves significantly better accuracy. Note that the memory requirement exceeds the model size (7B) because we use a batch size of 16 and a maximum token length of 600.

reduced. While MaZO introduces a small memory overhead, its benefits in terms of faster convergence and reduced gradient variance outweigh this cost, making it an effective and practical solution for multi-task fine-tuning under ZO optimization.

4.6 Various Weight Importance Metrics

To further validate the effectiveness of MaZO, we compare its performance with three alternative weight scoring methods: random selection, magnitude-based scoring, and Wanda scoring. Detailed implementation of these methods is described in Appendix F. For a fair comparison, we fix the sparsity level at 50%, consistent with the sparsity used in the Wanda score. Table 5 summarizes the results of this comparison.

The findings indicate that while both the magnitude-based and Wanda-based scoring can improve average performance, their improvements are less pronounced and less balanced across tasks compared to MaZO. This is because these methods evaluate the weight importance statically, without considering training dynamics or perturbation-based insights. In contrast, MaZO dynamically identifies critical parameters during training, enabling more effective optimization and better multi-task balance under the ZO framework. These results underscore the superiority of MaZO in leveraging weight importance to achieve state-of-the-art performance in multi-task fine-tuning.

Task	SST-2	BoolQ	Copa	SQuAD	Avg
No Mask	85.4	72.2	80.0	66.0	75.9
Random	86.6	73.0	80.0	63.4	75.8
Magnitude	87.4	75.6	79.0	65.6	76.9
Wanda	88.4	77.8	80.0	62.4	77.2
MaZO	**90.2**	**78.0**	**81.0**	**72.3**	**80.4**

Table 5: Comparison of different weight importance metrics. The sparsity is set to 50% except for *No Mask*. Random and Magnitude are done weight-wise while Wanda and MaZO are selected row-wise.

5 Conclusion

In this work, we have presented MaZO, a novel framework that harnesses masked zeroth-order optimization for the multi-task fine-tuning of LLMs. By incorporating weight importance score alongside a multi-task weight update mask, MaZO effectively reduces gradient variance and mitigates conflicts among tasks. Our experimental results demonstrate that MaZO not only surpasses current zeroth-order optimization methods but also outperforms leading multi-task learning methods designed for first-order optimization across a range of NLP tasks. Furthermore, our parameter-level approach is not limited solely to zeroth-order optimization, offering potential integrations with a variety of other optimization strategies.

6 Limitations

While MaZO demonstrates strong empirical performance, several limitations warrant discussion. First, the computation of weight importance introduces additional computational overhead compared to vanilla ZO methods. However, this cost remains negligible relative to the memory and computational demands of model weights and activations. Second, the effectiveness of MaZO is partially contingent on the quality of gradient and Hessian approximations. While our current approximations are effective, they could be further refined through more sophisticated estimation techniques to enhance performance. Finally, we do not provide a theoretical convergence analysis specifically for the MaZO approach. However, Sparse MeZO (Liu et al., 2024b) has already conducted a comprehensive and rigorous analysis of general masking scenarios in zeroth-order optimization. We refer interested readers to their work for detailed theoretical insights, and therefore do not duplicate these efforts here.

References

Emad Aghajanzadeh, Tahereh Bahraini, Amir Hossein Mehrizi, and Hadi Sadoghi Yazdi. 2023. Task weighting based on particle filter in deep multi-task learning with a view to uncertainty and performance. *Pattern Recognition*, 140:109587.

Arash Ahmadian, Seraphina Goldfarb-Tarrant, Beyza Ermis, Marzieh Fadaee, Sara Hooker, et al. 2024. Mix data or merge models? optimizing for diverse multi-task learning. *arXiv preprint arXiv:2410.10801*.

Anthropic. 2024. Claude 3.5 sonnet. Available at `https://www.anthropic.com/claude/sonnet`.

Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. 2024a. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tianyi Bai, Hao Liang, Binwang Wan, Yanran Xu, Xi Li, Shiyu Li, Ling Yang, Bozhou Li, Yifan Wang, Bin Cui, et al. 2024b. A survey of multimodal large language model from a data-centric perspective. *arXiv preprint arXiv:2405.16640*.

Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. 2018. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. 2023. Deepzero: Scaling up zeroth-order optimization for deep model training. *arXiv preprint arXiv:2310.02025*.

Shijie Chen, Yu Zhang, and Qiang Yang. 2024a. Multi-task learning in natural language processing: An overview. *ACM Comput. Surv.*, 56(12).

Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. 2019. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. *Advances in neural information processing systems*, 32.

Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. 2024b. Enhancing zeroth-order fine-tuning for language models with low-rank structures. *arXiv preprint arXiv:2410.07698*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. 2023. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*.

Google DeepMind. 2024. Gemini 2.0. Available at `https://deepmind.google/technologies/gemini/`.

Jean-Antoine Désidéri. 2012. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint arXiv:2403.03432*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. 2024. Variance-reduced zeroth-order methods for fine-tuning language models. *arXiv preprint arXiv:2404.08080*.

Zi Gong, Hang Yu, Cong Liao, Bingchang Liu, Chaoyu Chen, and Jianguo Li. 2024. Coba: Convergence balancer for multitask finetuning of large language models. *arXiv preprint arXiv:2410.06741*.

Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, et al. 2024. Zeroth-order fine-tuning of llms with extreme sparsity. *arXiv preprint arXiv:2406.02913*.

Shashank Gupta, Subhabrata Mukherjee, Krishan Subudhi, Eduardo Gonzalez, Damien Jose, Ahmed H Awadallah, and Jianfeng Gao. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *arXiv preprint arXiv:2204.07689*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Shuoran Jiang, Qingcai Chen, Youcheng Pan, Yang Xiang, Yukang Lin, Xiangping Wu, Chuanyi Liu, and Xiaobao Song. 2024. Zo-adamu optimizer: Adapting perturbation by the momentum and uncertainty in zeroth-order optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18363–18371.

Sarawoot Kongyoung, Craig Macdonald, and Iadh Ounis. 2020. Multi-task learning using dynamic task weighting for conversational question answering. In *Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI)*, pages 17–26, Online. Association for Computational Linguistics.

David Kozak, Cesare Molinari, Lorenzo Rosasco, Luis Tenorio, and Silvia Villa. 2023. Zeroth-order optimization with orthogonal random directions. *Mathematical Programming*, 199(1):1179–1219.

Joyjit Kundu, Wenzhe Guo, Ali BanaGozar, Udari De Alwis, Sourav Sengupta, Puneet Gupta, and Arindam Mallik. 2024. Performance modeling and workload analysis of distributed large language model training and inference. In *2024 IEEE International Symposium on Workload Characterization (IISWC)*, pages 57–67. IEEE.

Zeman Li, Xinwei Zhang, Peilin Zhong, Yuan Deng, Meisam Razaviyayn, and Vahab Mirrokni. 2024. Addax: Utilizing zeroth-order gradients to improve memory efficiency and performance of sgd for fine-tuning language models. *arXiv preprint arXiv:2410.06441*.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. 2024a. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient finetuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.

Shengchao Liu, Yingyu Liang, and Anthony Gitter. 2019. Loss-balanced task weighting to reduce negative transfer in multi-task learning.

Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. 2020. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. 2024b. Sparse mezo: Less parameters for better performance in zeroth-order llm fine-tuning. *arXiv preprint arXiv:2402.15751*.

Debabrata Mahapatra and Vaibhav Rajan. 2020. Multitask learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning large language models with just forward passes.

Yuren Mao, Zekai Wang, Weiwei Liu, Xuemin Lin, and Pengtao Xie. 2022. MetaWeighting: Learning to weight tasks in multi-task learning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3436–3448, Dublin, Ireland. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4o. Available at `https://www.openai.com/gpt-4o`.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Zhyar Rzgar K Rostam, Sándor Szénási, and Gábor Kertész. 2024. Achieving peak performance for large language models: A systematic review. *IEEE Access*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. 2023. Recon: Reducing conflicting gradients from the root for multi-task learning. *arXiv preprint arXiv:2302.11289*.

James C Spall. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Fei Wang, Li Shen, Liang Ding, Chao Xue, Ye Liu, and Changxing Ding. 2024. Simultaneous computation and memory efficient zeroth-order optimizer for fine-tuning large language models. *arXiv preprint arXiv:2410.09823*.

Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. 2023. Multilora: Democratizing lora for better multi-task learning. *arXiv preprint arXiv:2311.11501*.

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, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Denvy Deng, Feng Sun, Qi Zhang, et al. 2024b. Mtl-lora: Low-rank adaptation for multi-task learning. *arXiv preprint arXiv:2410.09437*.

Yifan Yang, Kai Zhen, Ershad Banijamal, Athanasios Mouchtaris, and Zheng Zhang. 2024c. Adazeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. *arXiv preprint arXiv:2406.18060*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*.

Ziming Yu, Pan Zhou, Sike Wang, Jia Li, and Hua Huang. 2024. Subzero: Random subspace zeroth-order optimization for memory-efficient llm fine-tuning. *arXiv preprint arXiv:2410.08989*.

Qi Zhang, Peiyao Xiao, Kaiyi Ji, and Shaofeng Zou. 2024a. On the convergence of multi-objective optimization under generalized smoothness. *arXiv preprint arXiv:2405.19440*.

Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D Lee, Wotao Yin, Mingyi Hong, et al. 2024b. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. *arXiv preprint arXiv:2402.11592*.

Zhihan Zhang, Wenhao Yu, Mengxia Yu, Zhichun Guo, and Meng Jiang. 2023. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 943–956, Dubrovnik, Croatia. Association for Computational Linguistics.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W Tsang. 2024. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. *arXiv preprint arXiv:2402.15173*.

Yequan Zhao, Xinling Yu, Zhixiong Chen, Ziyue Liu, Sijia Liu, and Zheng Zhang. 2023. Tensor-compressed back-propagation-free training for (physics-informed) neural networks. *arXiv preprint arXiv:2308.09858*.

Xiaojun Zhou, Yuan Gao, Chaojie Li, and Zhaoke Huang. 2022. A multiple gradient descent design for multi-task learning on edge computing: Multi-objective machine learning approach. *IEEE Transactions on Network Science and Engineering*, 9(1):121–133.

Yan Zhuang, Zhenzhe Zheng, Fan Wu, and Guihai Chen. 2024. Litemoe: Customizing on-device llm serving via proxy submodel tuning. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, SenSys '24, page 521–534, New York, NY, USA. Association for Computing Machinery.

A Additional Explanation on Hessian and Gradient Approximation

Consider a linear layer in an LLM that computes:

$$\mathbf{y} = \mathbf{W}\mathbf{x}, \tag{17}$$

where $\mathbf{W} \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{y} \in \mathbb{R}^m$. Focusing on one particular linear component, let us analyze a single row $\mathbf{w}_i \in \mathbb{R}^n$ of $\mathbf{W}$. The corresponding output is given by:

$$\mathbf{y}_i = \mathbf{w}_i\,\mathbf{x}, \tag{18}$$

which is a scalar.

To analyze the sensitivity of the loss $\mathcal{L}$ with respect to $\mathbf{w}_i$, we perform a second-order Taylor expansion of $\mathcal{L}$ with respect to y_i:

$$\begin{aligned}\mathcal{L}(\mathbf{y}_i + \Delta\mathbf{y}_i) &\approx \mathcal{L}(\mathbf{y}_i) + \nabla\mathcal{L}(\mathbf{y}_i)\,\Delta\mathbf{y}_i \\ &\quad + \frac{1}{2}\nabla^2\mathcal{L}(\mathbf{y}_i)\,(\Delta\mathbf{y}_i)^2. \end{aligned} \tag{19}$$

Since $\mathbf{y}_i$ is a scalar, its second derivative $\nabla^2\mathcal{L}(\mathbf{y}_i)$ is also a scalar.

Now, the change in $\mathbf{y}_i$ due to a change in the weights is

$$\Delta\mathbf{y}_i = \Delta\mathbf{w}_i\,\mathbf{x}. \tag{20}$$

Substituting this into the second-order term yields:

$$\frac{\partial^2\mathcal{L}}{\partial\mathbf{w}_i^2} \approx \nabla^2\mathcal{L}(\mathbf{y}_i)\,(\mathbf{x}\mathbf{x}^\top). \tag{21}$$

Since we are primarily interested in comparing weight importance along the row direction, the absolute scale of the Hessian is not crucial. In practice, we can drop the multiplicative factor $\nabla^2\mathcal{L}(y_i)$ (or, equivalently, assume it to be a constant) and write:

$$\frac{\partial^2\mathcal{L}}{\partial\mathbf{w}_i^2} \propto \mathbf{x}\mathbf{x}^\top. \tag{22}$$

Similarly, one can derive a first-order approximation for the gradient. By retaining only the first-order term of the Taylor expansion, we have:

$$\mathcal{L}(\mathbf{y}_i + \Delta\mathbf{y}_i) \approx \mathcal{L}(\mathbf{y}_i) + \nabla\mathcal{L}(\mathbf{y}_i)\,\Delta\mathbf{y}_i. \tag{23}$$

With $\Delta\mathbf{y}_i = \Delta\mathbf{w}_i\,\mathbf{x}$, the gradient with respect to $\mathbf{w}_i$ becomes:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}_i} \approx \nabla\mathcal{L}(\mathbf{y}_i)\,\mathbf{x}. \tag{24}$$

Similarly, since we are only interested in the relative value, the factor is dropped:

$$\mathbf{g} \propto \mathbf{x}. \tag{25}$$

Algorithm 1 MaZO LLM Fine-Tuning Framework

```
**Input:** Pre-trained LLM parameters $\theta$, training data, tasks $t = 1, \ldots, T$, sparsity level $\rho$, hyperparameters $\alpha$, $\beta$, learning rate $\eta$
**Output:** Updated parameters $\theta^*$
**for** each task $t = 1$ to $T$ **do**
    Collect evaluation data
    **Compute** $\mathbf{S}^t_{\text{global}}$ with eq. (12)
    **Compute** $\mathbf{S}^t_{\text{greedy}}$ with eq. (14)
    **Combine** scores:
        $\mathbf{S}^t = \mathbf{S}^t_{\text{global}} + \alpha\, \mathbf{S}^t_{\text{greedy}} + \beta|\mathbf{W}|$
    **Normalize** $\mathbf{S}^t$ and get $\hat{\mathbf{S}}^t$ with eq. (6)
**end for**
**Aggregate:** Sum row-wise normalized scores across tasks:
    $\mathbf{S} = \sum_{t=1}^{T} \hat{\mathbf{S}}^t$
**for** each weight $\mathbf{W}$ in LLM **do**
    **for** each row $i$ in $\mathbf{W}$ **do**
        Select top $k$ parameters in each row according to corresponding $\mathbf{S}$ and construct weight update mask $\mathbf{M}$
    **end for**
**end for**
**for** each training step **do**
    **Compute** weight update $\Delta\mathbf{W}$ using ZO optimization
    **Apply mask:**
        $\Delta\mathbf{W}_{\text{masked}} = \Delta\mathbf{W} \odot \mathbf{M}$
    **Update** parameters:
        $\theta \leftarrow \theta + \eta\Delta\mathbf{W}_{\text{masked}}$
**end for**
```

This derivation shows that, by considering each row independently (row-wise), we avoid the immense complexity involved in computing the full Hessian matrix (which is high-dimensional and difficult to characterize even under diagonalization assumptions). In other words, computing the Hessian row-wise allows us to circumvent the problem of determining the eigenvalues or even a reliable diagonal approximation of the full Hessian.

B Implementation Details and Reproducibility

B.1 Hyperparameters

The training configuration involves setting a learning rate of 1e-7 for the full model and 3e-4 for the LoRA-based components. The batch size is configured to 8, and the training process consists of 30,000 steps. The optimizer used is Stochastic Gradient Descent (SGD). In terms of sparsity, a MaZO sparsity of 0.9 is applied to the full model, while 0.8 sparsity is utilized for the LoRA components. Additionally, the LoRA rank is defined as 16.

B.2 Ablation Study

We explore the optimal hyperparameter settings for MaZO, includeing α, β, sparsity level, and the LoRA rank. To streamline the process, we perform grid searches for each hyperparameter while keeping the others constant. For most experiments, we fine-tune the model on SST-2, BoolQ, COPA, and SQuAD, encompassing binary classification, multiple-choice, and generation tasks, providing diverse evaluation scenarios. However, for the LoRA rank, we evaluate performance across all tasks.

α and β. To optimize α and β, we fix the sparsity level at 50% and perform full-model fine-tuning (without LoRA). The search is conducted in two stages. First, β is set to zero, and α is tuned, resulting in an optimal value of $\alpha = 10$. Next, with α fixed, β is tuned, yielding an optimal value of $\beta = 1$. These values strike a balance between the global and greedy weight importance metrics, ensuring effective parameter selection.

LoRA rank. We examine the impact of LoRA rank and provide detailed results in Appendix E. In summary, the results reveal a U-shaped relationship between rank and performance, reflecting a trade-off between model capacity and dimensionality. The optimal rank of 16 minimizes loss and is used as the default setting for LoRA-based baseline.

Sparsity. We perform a grid search of the sparsity level ρ from 0.1 to 0.99. For full-model fine-tuning, the performance first improves with increasing sparsity and then sharply declines. The peak performance is achieved at $\rho = 0.9$. For LoRA fine-tuning, we jointly optimize sparsity levels and LoRA ranks. The optimal result is found at a LoRA rank of 64 and a sparsity level of 0.8. Notably, the effective number of parameters is equivalent to $64 \times (1 - 0.8) = 12.8$, which is less than the best-performing rank of LoRA baseline. This highlights that MaZO can further reduce the dimension while maintaining the model capacity.

C Pseudo-code of MaZO

The pseudo-code of the whole MaZO LLM fine-tuning framework is shown as Algorithm 1.

D Baseline

D.1 CoBa: Convergence Balancer for Multitask Finetuning

CoBa (Convergence Balancer) (Gong et al., 2024) is a novel multi-task learning (MTL) method designed for large language models (LLMs). It dynamically adjusts task weights during training to ensure balanced convergence across tasks, while maintaining computational efficiency.

Consider an LLM parameterized by $\theta \in \mathbb{R}^m$, trained on $T \geq 2$ tasks. The loss function for task t at iteration i is denoted as $\mathcal{L}^t(\theta; i) : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$. The overall optimization objective is:

$$\min_{\theta \in \mathbb{R}^m} \mathcal{L}(\theta; i) = \sum_{t=1}^{T} \omega_t(i) \mathcal{L}^t(\theta; i), \tag{26}$$

where $\omega_t(i)$ is the weight of task t at iteration i. A uniform weight assignment $\omega_t(i) = \frac{1}{T}$ ensures equal attention to all tasks but often leads to varying convergence rates. CoBa dynamically adjusts $\omega_t(i)$ to balance these rates, prioritizing generalization by deriving weights from validation losses instead of training losses. CoBa is built upon three main components:

Relative Convergence Score (RCS) dynamically allocates smaller weights to tasks that converge faster and larger weights to slower-converging tasks. It is computed as:

$$\text{RCS}_t(i) = \text{softmax}_t \left(T \frac{\alpha_t(i)}{\sum_{t'=1}^{T} |\alpha_{t'}(i)|} \right), \tag{27}$$

where $\alpha_t(i)$ is the convergence slope of task t, derived from the normalized validation loss ratio over a sliding window of N iterations. The softmax operation ensures differentiation across tasks, with faster-converging tasks receiving lower weights.

Absolute Convergence Score (ACS) addresses task divergence by reducing weights for diverging tasks and increasing weights for converging tasks. It is computed as:

$$\text{ACS}_t(i) = \text{softmax}_t \left(-N \frac{\alpha_t(i)}{\sum_{j=i-N+1}^{i} |\alpha_t(j)|} \right), \tag{28}$$

where normalization is performed along the historical iteration dimension, isolating a task's own trajectory. ACS ensures tasks with consistent convergence receive higher weights while diverging tasks are penalized.

Divergence Factor (DF) determines the relative influence of RCS and ACS on the final task weights. It is defined as:

$$\text{DF}(i) = \min \left(\text{softmax}_i \left(\frac{i \cdot \alpha_{\text{max}}(i)}{\sum_{j=1}^{i} \alpha_{\text{max}}(j)} \right), 1 \right), \tag{29}$$

where $\alpha_{\text{max}}(i)$ is the largest convergence slope across all tasks at iteration i. DF ensures RCS dominates when all tasks are converging, while ACS takes precedence when divergence is detected.

The final task weights $\omega_t(i)$ are computed as:

$$\omega_t(i) = \text{DF}(i) \cdot \text{RCS}_t(i) + (1 - \text{DF}(i)) \cdot \text{ACS}_t(i), \tag{30}$$

allowing a seamless transition between RCS and ACS dominance based on task convergence trends.

The convergence slope $\alpha_t(i)$ for task t is calculated based on the normalized validation loss ratio $\bar{\mathcal{L}}_t^{\text{val}}(\theta; i)$. Specifically, we fit a linear model to the validation loss ratios over a sliding window of N iterations. The observations are defined as:

$$\mathbf{x}_t(i) = [i, 1]^\top \tag{31}$$

$$\mathbf{X}_t(N; i) = [\mathbf{x}_t(s_0), \ldots, \mathbf{x}_t(i)]^\top \tag{32}$$

$$\mathbf{y}_t(N; i) = [\bar{\mathcal{L}}_t^{\text{val}}(\theta; s_0), \ldots, \bar{\mathcal{L}}_t^{\text{val}}(\theta; i)]^\top \tag{33}$$

where $s_0 = \max(0, i - N + 1)$ is the starting step of the sliding window. The goal is to compute the coefficient vector $c_t(N; i) = [\alpha_t(N; i), \beta_t(N; i)]^\top$ that minimizes the mean squared error (MSE) between the predicted and actual validation loss ratios:

$$c_t = \arg\min_{c_t} \frac{1}{2} (\mathbf{X}_t c_t - \mathbf{y}_t)^\top (\mathbf{X}_t c_t - \mathbf{y}_t). \tag{34}$$

The closed-form solution for c_t is given by:

$$c_t = (\mathbf{X}_t^\top \mathbf{X}_t)^{-1} \mathbf{X}_t^\top \mathbf{y}_t. \tag{35}$$

Algorithm The CoBa algorithm is summarized in Algorithm 2, We use $M = 4$ with batchsize $= 16$

Algorithm 2 CoBa Algorithm

```
Require: Initial parameters $\theta_0$, $M$ batches of validation set, history window length $N = 5M$, warm-up steps $W = M$, number of tasks $T$, initial weights $\omega_i(0) = \frac{1}{T}$.
Ensure: Trained parameters $\theta$.
 1: for $t = 0$ to $T$ do
 2:     Compute $\mathcal{L}(\theta; i)$ with training batch $x_i$.
 3:     Compute $\bar{\mathcal{L}}_t^{\text{val}}(\theta; t)$ with validation batch $v_i$.
 4:     Update validation loss history $\mathbf{y}_t(N; i)$.
 5:     Compute $\alpha_t(i)$.
 6:     if $i > W$ then
 7:         Compute RCS($i$), ACS($i$), and DF($i$) using Eqs. (27), (28), and (29).
 8:         Update task weights $\omega_t(i)$ using Eq. (30).
 9:     else
10:         Set $\omega_t(i) = \frac{1}{T}$.
11:     end if
12:     Update model parameters $\theta$ using weighted loss $\mathcal{L}(\theta; i)$.
13: end for
```

D.2 FAMO: Fast Adaptive Multitask Optimization

Fast Adaptive Multitask Optimization (FAMO) is a dynamic weighting method designed to address the challenges of multitask learning (MTL), where directly optimizing the average loss across tasks often leads to under-optimization of certain tasks. FAMO ensures balanced task loss reduction using only $O(1)$ space and time per iteration, making it computationally efficient and scalable.

The complete FAMO algorithm is summarized in Algorithm 4.

D.3 MTL-LoRA

MTL-LoRA (Multi-Task Learning LoRA) is a parameter-efficient fine-tuning method designed to enhance the multi-task learning (MTL) capabilities of large language models (LLMs). It builds upon the Low-Rank Adaptation (LoRA) framework by addressing the challenges of task interference and suboptimal information sharing in multi-task scenarios.

LoRA is a parameter-efficient fine-tuning method that freezes the majority of a pre-trained model's parameters and introduces trainable low-rank matrices to approximate gradient updates. For a weight matrix $\mathbf{W} \in \mathbb{R}^{d\times k}$ in the original model,

Algorithm 3 PyTorch Implementation of Wanda

```
Input: Weight matrix $\mathbf{W} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$, input activations $\mathbf{X} \in \mathbb{R}^{(N\cdot L)\times C_{\text{in}}}$, sparsity ratio $s \in [0, 1]$
Output: Pruned weight matrix $\mathbf{W}$
Compute importance scores: metric $= \mathbf{W}$.abs() $\cdot$ $\mathbf{X}$.norm($p = 2$, dim $= 0$)
Sort scores within each row: _, sorted_idx $=$ torch.sort(metric, dim $= 1$)
Identify indices to prune: pruned_idx $=$ sorted_idx[:, : $\lfloor C_{\text{in}} \cdot s \rfloor$]
Set pruned weights to zero: $\mathbf{W}$.scatter_(dim $=$ 1, index $=$ pruned_idx, src $= 0$)
Return W
```

LoRA decomposes the gradient update ΔW into two low-rank matrices $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$, where $r \ll \min(d, k)$. The updated weight matrix is expressed as:

$$\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \mathbf{BA}.$$

The output of the updated layer for an input x is:

$$\mathbf{h} = (\mathbf{W} + \mathbf{BA})\mathbf{x}.$$

MTL-LoRA enhances LoRA by introducing task-specific transformations and dynamic information-sharing strategies.

Task-Specific Transformation. MTL-LoRA introduces a learnable task-specific transformation matrix $\mathbf{\Lambda}_t \in \mathbb{R}^{r\times r}$ for each task t. For an input $\mathbf{x}_t$ corresponding to task t, the low-rank projection is modified as:

$$\mathbf{z}_t = \mathbf{\Lambda}_t \mathbf{A} \mathbf{x}_t,$$

where $\mathbf{A} \in \mathbb{R}^{r\times k}$ is the shared low-rank matrix.

Dynamic Information Sharing. To improve cross-task information sharing, MTL-LoRA employs multiple up-projection matrices $\mathbf{B}_i \in \mathbb{R}^{d\times r}$ $(i = 1, \ldots, n)$ and combines their outputs using a weighted averaging strategy. The final output for task t is computed as:

$$\mathbf{h}_t = \mathbf{W}\mathbf{x}_t + \sum_{i=1}^{n} \frac{\exp(w_i^t/\tau)}{\sum_{j=1}^{n}\exp(w_j^t/\tau)} \mathbf{B}_i \mathbf{z}_t,$$

where w_i^t are learnable weights for task t, and τ is a temperature hyperparameter controlling the sharpness of the weight distribution.

We set number of up-projection matrices n to 3, rank to 16 and temperature τ to 0.5

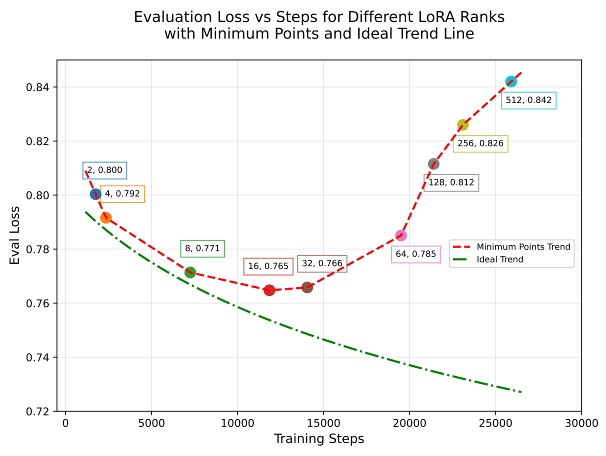

Figure 5: The loss curve with different LoRA rank.

E LoRA Rank

We investigate the influence of LoRA rank on the model's final performance. Initially, we exclude weight masking and fine-tune the model with different LoRA ranks. The evaluation loss curves for ranks ranging from 1 to 512 are plotted in Figure 5. As the rank increases, the loss forms a U-shaped curve, with the lowest point occurring at a rank of 16. Ideally, the trend of the lowest point in first-order (FO) optimization should follow the green dashed line in Figure 5. However, in the zeroth-order (ZO) setting, the larger parameter optimization space as rank increases leads to a deviation from this ideal trend.

This U-shaped curve highlights a critical trade-off: while increasing the rank improves the model's capacity, it simultaneously introduces challenges in optimizing a larger parameter space under ZO settings. This observation directly motivates our exploration of sparsity and mask selection strategies, which aim to reduce the number of parameters being optimized while retaining the most important ones. By identifying and focusing on the most critical parameters, we can mitigate the challenges posed by ZO optimization and achieve better performance, as demonstrated by our MaZO approach.

F Details of Different Weight Score Metrics

F.1 Wanda: Pruning by Weights and Activations

In this section, we introduce **Wanda** (Pruning by Weights and Activations), a simple yet effective method for pruning large language models (LLMs). Wanda can induce high sparsity in pretrained LLMs without requiring retraining or weight updates, making it computationally efficient and easy to implement.

The key idea of Wanda is to evaluate the importance of each weight based on both its magnitude and the corresponding input activation. Specifically, for a linear layer with weight matrix $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$ and input activations $\mathbf{X} \in \mathbb{R}^{(N \cdot L) \times C_{\text{in}}}$, the importance score $\mathbf{S}_{ij}$ of weight $\mathbf{W}_{ij}$ is defined as:

$$\mathbf{S}_{ij} = |\mathbf{W}_{ij}| \cdot \|\mathbf{X}_j\|_2, \tag{36}$$

where $|\mathbf{W}_{ij}|$ is the absolute value of the weight, and $\|\mathbf{X}_j\|_2$ is the $\mathcal{L}_2$ norm of the j-th column of $\mathbf{X}$, aggregated across all tokens in the batch and sequence dimensions. This metric effectively combines weight magnitude and input activation information to determine the importance of each weight.

Unlike traditional pruning methods that compare weights globally or layer-wise, Wanda adopts a per-output comparison strategy (the same as our row-wise comparison). For a weight $\mathbf{W}_{ij}$ connecting input j to output i, its comparison group is defined as all weights connected to the same output i:

$$\mathbf{G}_{ij} = \{\mathbf{W}_{uv} \,|\, u = i\}. \tag{37}$$

Within each comparison group, weights are ranked by their importance scores $\mathbf{S}_{ij}$, and a predefined sparsity ratio $s\%$ is applied to prune the lowest-ranked weights.

F.2 Other Metrics

In this section, we introduce two additional heuristic weight importance metrics: random and magnitude.

For the random metric, we randomly select 50% of the weights. It is important to note that the comparison group is the entire set of weights, rather than a single row.

For the magnitude metric, we select weights with the smallest values in a weight, following the approach described by Liu et al. (2024b).

G Task Details

We consider a diverse set of natural language understanding (NLU) and natural language generation (NLG) tasks.

G.1 Natural Language Understanding Tasks

We select tasks from the GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) benchmarks:

- **SST-2** (Stanford Sentiment Treebank): A binary sentiment classification task.
- **BoolQ**: A yes/no question-answering task.
- **RTE** (Recognizing Textual Entailment): A binary classification task for textual entailment.
- **WSC** (Winograd Schema Challenge): A pronoun resolution task.
- **WiC** (Word-in-Context): A word sense disambiguation task.
- **MultiRC** (Multi-Sentence Reading Comprehension): A question-answering task where each question has multiple correct answers.
- **COPA** (Choice of Plausible Alternatives): A multiple-choice task for causal reasoning.

G.2 Natural Language Generation Task

For natural language generation, we include:

- **SQuAD** (Rajpurkar, 2016): A question-answering dataset where the model generates text-based answers from a given passage.

G.3 Dataset Splits and Evaluation Metrics

To ensure computational feasibility, we randomly sample 500 instances for training, 250 for validation, and 500 for testing for each task. Performance is measured using F1 score or accuracy, depending on the task.

H Discussion of Collinearity

Task-specific ZO gradients: For each task $t \in \{1, \ldots, T\}$, the zeroth-order gradient estimate is given by

$$\mathbf{g}^t = \frac{\mathcal{L}^t(\theta + \epsilon\mathbf{z}) - \mathcal{L}^t(\theta - \epsilon\mathbf{z})}{2\epsilon}\mathbf{z} \equiv \alpha_t\mathbf{z}, \quad (38)$$

where α_t is a scalar. Thus, every $\mathbf{g}^t$ is a scalar multiple of the same random direction $\mathbf{z}$.

Span of all task gradients: The space spanned by the set of all task gradients is

$$\text{span}\{\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^T\} = \text{span}\{\mathbf{z}\}. \quad (39)$$

Therefore, the dimension of this span is

$$\dim\left(\text{span}\{\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^T\}\right) = 1. \quad (40)$$

Aggregated gradient: The combined gradient used for the update is

$$\mathbf{g} = \sum_{t=1}^{T} w_t\mathbf{g}^t = \left(\sum_{t=1}^{T} w_t\alpha_t\right)\mathbf{z}, \quad (41)$$

which clearly lies in the one-dimensional subspace spanned by $\mathbf{z}$.

Gradient covariance matrix: Define the covariance matrix of the task gradients as

$$\mathbf{C} = \sum_{t=1}^{T} \pi_t \left(\mathbf{g}^t - \bar{\mathbf{g}}\right)\left(\mathbf{g}^t - \bar{\mathbf{g}}\right)^\top, \quad (42)$$

where π_t are probability weights (or simply $1/T$ for uniform weighting) and the mean gradient is

$$\bar{\mathbf{g}} = \sum_{t=1}^{T} \pi_t\mathbf{g}^t. \quad (43)$$

Since $\mathbf{g}^t = \alpha_t\mathbf{z}$, we have

$$\mathbf{g}^t - \bar{\mathbf{g}} = (\alpha_t - \bar{\alpha})\mathbf{z}, \quad \text{with } \bar{\alpha} = \sum_{t=1}^{T} \pi_t\alpha_t. \quad (44)$$

Thus, the covariance matrix becomes

$$\mathbf{C} = \left(\sum_{t=1}^{T} \pi_t(\alpha_t - \bar{\alpha})^2\right)\mathbf{z}\mathbf{z}^\top. \quad (45)$$

Since $\mathbf{z}\mathbf{z}^\top$ is an outer product of a vector with itself, it has rank 1. Hence,

$$\text{rank}(\mathbf{C}) = 1. \quad (46)$$

Conclusion: The lack of *directional diversity* in the task gradients is mathematically captured by the fact that all task-specific gradients lie in a one-dimensional subspace, and the covariance matrix of these gradients has rank 1. This indicates that no matter how many tasks are aggregated, the update direction remains confined to a single direction $\mathbf{z}$ in the parameter space.

Algorithm 4 Fast Adaptive Multitask Optimization (FAMO)

```
Require: Initial model parameters $\theta_0$, task losses $\{\mathcal{L}_{t,i}\}_{t=1}^{T}$, learning rates $\alpha$ and $\beta$, decay factor $\gamma$.
1: Initialize logits: $\xi_1 \leftarrow \mathbf{0}$.
2: for $i = 1$ to $T$ do
3:     Compute task weights:
           $$\mathbf{z}_i = \mathrm{Softmax}(\xi_t),$$
       where for each $i$,
           $$\mathbf{z}_{t,i} = \frac{\exp(\xi_{t,i})}{\sum_{t'=1}^{T}\exp(\xi_{t',i})}.$$
4:     Update model parameters:
           $$\theta_{t+1} = \theta_t - \alpha \sum_{t=1}^{T} \left( c_t \frac{\mathbf{z}_{t,i}}{\mathcal{L}_{t,i}} \right) \nabla \mathcal{L}_{t,i},$$
           $$\text{with} \quad c_i = \left( \sum_{i=1}^{k} \frac{\mathbf{z}_{t,i}}{\mathcal{L}_{t,i}} \right)^{-1}.$$
5:     Compute the vector of log-loss differences:
           $$d_i = \begin{bmatrix} \log \mathcal{L}_{1,i} - \log \mathcal{L}_{1,i+1} \\ \vdots \\ \log \mathcal{L}_{T,i} - \log \mathcal{L}_{T,i+1} \end{bmatrix}.$$
6:     Compute the Jacobian of the softmax function:
           $$(J_i)_{tt'} = \frac{\partial z_{t,i}}{\partial \xi_{t',i}} = z_{t,i}(\delta_{tt'} - z_{t',i}).$$
7:     Aggregate the gradient by the chain rule:
           $$\delta_i = J_i^{\top} d_i.$$
8:     Update logits:
           $$\xi_{i+1} = \xi_i - \beta\big(\delta_i + \gamma\, \xi_i\big).$$
9: end for
```