MetaGPT: Merging Large Language Models Using Model Exclusive Task Arithmetic

Anonymous ACL submission

Abstract

The advent of large language models (LLMs) like GPT-4 has catalyzed the exploration of multi-task learning (MTL), in which a single model demonstrates proficiency across diverse tasks. Task arithmetic has emerged as a cost-effective approach for MTL. It enables performance enhancement across multiple tasks by adding their corresponding task vectors to a pre-trained model. However, the current lack of a method that can simultaneously achieve optimal performance, computational efficiency, and data privacy limits their application to LLMs. In this paper, we propose **M**odel **E**xclusive **T**ask **A**rithmetic for merging **GPT**-scale models (`MetaGPT`), which formalizes the objective of model merging into a multi-task learning framework, aiming to minimize the average loss difference between the merged model and each individual task model. Since data privacy limits the use of multi-task training data, we leverage LLMs' local linearity and task vectors' orthogonality to separate the data term and scaling coefficients term and derive a model-exclusive task arithmetic method. Our proposed `MetaGPT` is data-agnostic and bypasses the heavy search process, making it cost-effective and easy to implement for LLMs. Extensive experiments demonstrate that `MetaGPT` leads to improvements in task arithmetic and achieves state-of-the-art performance on multiple tasks.

1 Introduction

In recent years, a well-established paradigm for AI has been to pre-train models using large-scale datasets and then to fine-tune the models on different tasks through supervised learning with task-specific datasets, which can lead to improved performance while requiring less labeled data (Devlin et al., 2018; OpenAI, 2023; Dodge et al., 2020). However, for each new application, a separate model has to be fine-tuned and deployed, which is computationally expensive and resource-intensive (Fifty et al., 2021; Zhang and Yang, 2021). Thus, Multi-Task Learning (MTL) methods have been proposed and developed to enable a single model to solve multiple tasks concurrently.

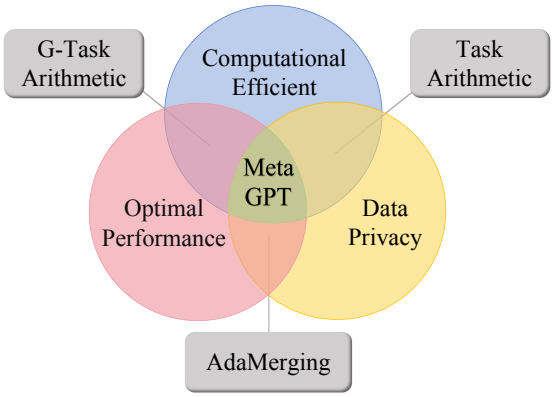

Figure 1: Existing methods face the trilemma of performance, data privacy, and computational costs, which hinders its application to LLMs. Our `MetaGPT` can solve these problems under careful approximation and thus can scale to GPT3-scale LLMs.

Conventional MTL approaches typically involve collecting raw data across multiple tasks and then jointly training a single model (Caruana, 1997; Yang et al., 2023a). However, the fine-tuning process becomes extremely computationally intensive with the development of large language models (LLMs) that may comprise billions or even trillions of parameters. Therefore, researchers have explored merging various task-specific models with the expectation that the merged model can handle multiple tasks simultaneously.

One of the outstanding merging methods is task arithmetic (Ilharco et al., 2023). For a given task, the element-wise difference between the weights of the pre-trained model and the fine-tuned model is referred to as the task vector. Recent studies have

shown that linearly adding multiple scaled task vectors to the pre-trained model can improve performance across those tasks (Ilharco et al., 2023; Yang et al., 2023b). Nevertheless, previous task arithmetic methods face a trilemma in practice. 1) The best-performing task arithmetic methods require extra training to obtain optimal hyper-parameters, but the high computational costs hinder their application to GPT3-scale LLMs. 2) Some training-free methods heuristically set the scaling coefficient to a constant (*e.g.*, 0.3), which is efficient but leads to sub-optimal performance. 3) Some methods conduct grid search on the training/validation set, which is sometimes impractical and faces the risk of data privacy concerns. In summary, as illustrated in Figure 1, there is essentially no task arithmetic method suitable for billion-scale models that perform satisfactorily in practice.

To address the aforementioned problems, in this paper, we propose `MetaGPT`: an *optimal* and *efficient* task arithmetic method for MTL *without any data* (**m**odel **e**xclusive **t**ask **a**rithmetic). We begin by providing a detailed theoretical analysis of the task loss difference and average loss difference introduced by the task arithmetic algorithm. Since we aim to choose parameters that minimize the average loss difference, we first separate the data term and scaling coefficients, which also establishes a performance upper bound for task arithmetic. After separating the scaling coefficients, the final result is quadratic for each scaling coefficient, leading to a closed-form solution that is simple and effective to implement.

The experimental results on the LLaMA-2 (Touvron et al., 2023) and Mistral (Jiang et al., 2023) series demonstrate that the `MetaGPT` approach is superior to previous merging methods on several tasks. `MetaGPT` provides an efficient avenue to optimally implement task arithmetic for large-scale multi-task learning (MTL) and push the frontiers of language model merging. To sum up, our contributions include:

1. We provide the mathematical formulation of the optimization objective for task arithmetic and the first theoretical analysis of the performance bound for task arithmetic.

2. To achieve efficient, optimal, and model-exclusive task arithmetic, we separate the data term and scaling coefficients in the optimization objective, which leads to a closed-form solution for the scaling coefficients.

3. Our `MetaGPT` is orthogonal to existing task vector-improving methods and can be integrated to achieve higher performance.

4. Extensive experiments demonstrate that our `MetaGPT` can improve task arithmetic and achieve state-of-the-art performance.

2 Related Work

Model Merging. Currently, model merging has been developed for multiple uses such as improving performance on a single target task (Izmailov et al., 2018; Wortsman et al., 2022), improving out-of-domain generalization (Ramé et al., 2023; Cha et al., 2021; Arpit et al., 2022), and improving the performance of multi-task learning (Ilharco et al., 2023; Yadav et al., 2024; Yu et al., 2023), which is the core focus of our research. The range of applications has led to a proliferation of methods to improve beyond simple parameter averaging. Fisher merging (Matena and Raffel, 2022) tries to weight the importance of individual models using Fisher Information Matrix and uses it to merge different models. RegMean (Jin et al., 2022) formulate the merging problem as a regression problem and leads to an optimal solution for linear models. Task Arithmetic (Ilharco et al., 2023) presents a method for merging models by adding task vectors to the pre-trained model to improve multi-task performance. Ties Merging (Yadav et al., 2024) and DARE (Yu et al., 2023) propose to refine the task vectors by resolving the interference and removing extremely redundant components. Ortiz-Jimenez et al. (2024) propose that fine-tuning the models in their tangent space can amplify weight disentanglement and lead to substantial performance improvements.

Multi-Task Learning. Multi-task learning is a powerful method for solving multiple correlated tasks simultaneously (Caruana, 1997). Current MTL works mainly focus on learning the shared representations from designing specific architecture (Misra et al., 2016; Sun et al., 2020) or using specific optimization methods (Sener and Koltun, 2018; Liu et al., 2021). The former focuses on learning the shared representation using different methods such as designing specific representation sharing module (Liu et al., 2019; Ding et al., 2021), learning to branch (Lu et al., 2017; Guo et al., 2020), and based selection criteria (Ma et al., 2018; Hazimeh et al., 2021). And the latter focuses on balancing multiple tasks from the perspectives of task training weights (Sener and Koltun, 2018; Liu

et al., 2019), gradient dominance (Chen et al., 2018; He et al., 2022; Yang et al., 2023a), and solving gradient conflicts (Yu et al., 2020; Chen et al., 2020; Liu et al., 2021). However, the conventional MTL approaches for collecting raw data across multiple tasks for joint training are not suitable for LLMs. The factors contributing to this issue are twofold: first, computational inefficiency due to the substantial computational costs associated with updating pre-trained models; second, a significant number of data proprietors are reluctant to disclose valuable or privacy-sensitive raw data.

3 Preliminaries

3.1 Notation

Let $f : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$ be a neural network taking inputs $\boldsymbol{x} \in \mathcal{X}$ and parameterized by a set of weights $\boldsymbol{\theta} \in \Theta$. We assume $\mathcal{X} \subseteq \mathbb{R}^p$, $\Theta \subseteq \mathbb{R}^m$ and $\mathcal{Y} \subseteq \mathbb{R}^q$. We consider fine-tuning a pre-trained model $f(\cdot, \boldsymbol{\theta}_0)$ on T different tasks, with each task t consisting of a triplet $(\mathcal{D}_t, \mathcal{L}_t, \boldsymbol{\theta}_t)$, where $\mathcal{D}_t = (\mathcal{D}_t^{\text{train}}, \mathcal{D}_t^{\text{val}}, \mathcal{D}_t^{\text{test}})$ is the training, validation and test data of task t, $\mathcal{L}_t$ is the loss function of task t, and $\boldsymbol{\theta}_t$ is the model parameters fine-tuned on task t based on the pre-trained weight $\boldsymbol{\theta}_0$.

3.2 Task Arithmetic

Let the *task vector* of task t be the difference between the fine-tuned and the pre-trained weights:

$$\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0. \tag{1}$$

Task arithmetic aims to solve the multi-task learning problem by directly adding the scaled task vectors to the pre-trained model weight $\boldsymbol{\theta}_0$:

$$\boldsymbol{\theta}_{\text{final}} = \boldsymbol{\theta}_0 + \sum_{i=1}^{T} \lambda_i \boldsymbol{\tau}_i \tag{2}$$

where λ_i is the scaling coefficient of task vector τ_i. As illustrated in Eq. 2, the task arithmetic introduces T hyper-parameters $\{\lambda_i | i = 1, \cdots, T\}$ and the choice of these scaling coefficients has a significant influence on the performance of the merged model. Thus, selecting the appropriate scaling coefficients for different task vectors remains a challenging problem.

3.3 Existing Methods

Earlier task arithmetic (Ilharco et al., 2023; Yadav et al., 2024) propose to perform a grid search (G-Task Arithmetic) on the validation set to choose the optimal scaling coefficients. However, as the number of tasks increases, exploring all the scaling coefficient combinations faces the curse of dimensionality. Therefore, to simplify the problem, they use the same value for multiple scaling coefficients, thereby reducing the computational complexity. In the absence of the training/validation data, they set $\lambda = 0.3$ as the default setting for dataless arithmetic. Moreover, Adamerging (Yang et al., 2023b) aims to autonomously learn the coefficients from unlabeled test samples using entropy minimization.

3.4 Scalability Challenges for LLMs

The methods mentioned above are not suitable for scaling to LLMs: The grid search method requires extra validation/training data, which faces the risk of data privacy concerns and the curse of dimensionality when the number of tasks increases. For instance, conducting a grid search for three hyper-parameters, each with a discretization interval of 0.01, would require 10^6 forward passes across the entire dataset. Setting a fixed value such as 0.3 for all the λ_i is time-efficient and can be applied to LLMs, but it leads to sub-optimal performance. Using test data input to unsupervised optimize these hyper-parameters can lead to an optimal solution but requires extra data and necessitates loading multiple models for training. This process is both time and memory consuming, making it challenging to apply to LLMs. For example, merging three LLMs requires loading three LLMs simultaneously to optimize, which is extremely costly. The statement above suggests that scaling up existing optimal task arithmetic to LLMs remains a challenging problem.

4 Our Proposed `MetaGPT`

4.1 Overview

To solve the problems above, we propose a new algorithm `MetaGPT`, based on careful approximations to a closed-form solution, which easily scales to giant models both in terms of runtime as well as performance while protecting data privacy. In this section, we state the motivation and optimization problem and solve it step by step. All proofs of lemmas and theorems are provided in the appendix.

4.2 `MetaGPT` Optimization Objective

Definition 1 (Single Task Loss Difference)**.** For the fine-tuned model $\boldsymbol{\theta}_i$ and the task arithmetic merged model $\boldsymbol{\theta}_{\text{final}}$. The Task Loss Difference in task t

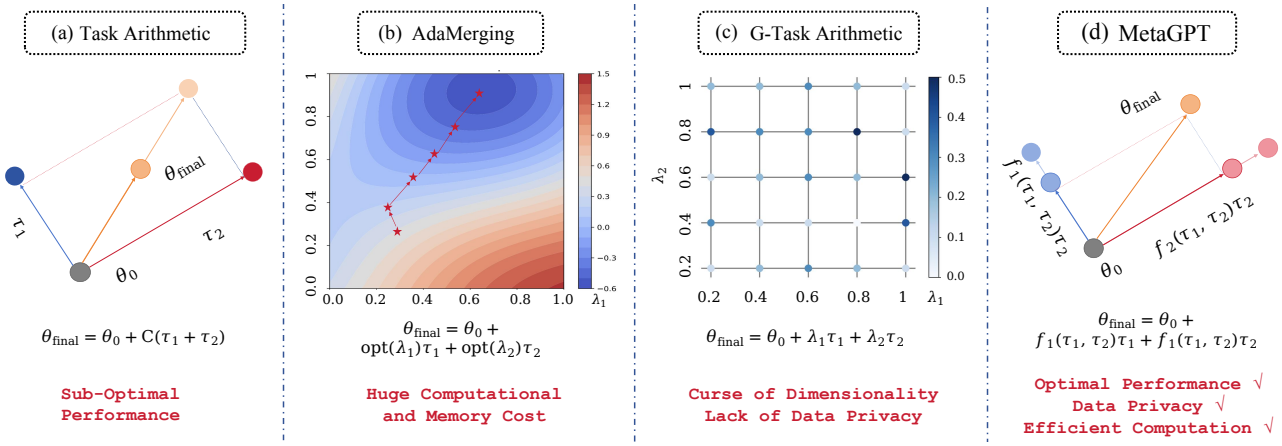

Figure 2: Current task arithmetic based methods face the problems of sub-optimal performance, huge computational and memory cost, curse of dimensionality and data privacy, which makes it difficult to scale to LLMs. Our method solves the aforementioned problems and provides an avenue to scale task arithmetic to LLMs.

(TLD_t) is defined as:

$$\text{TLD}_t(\lambda_1, \cdots, \lambda_T, \boldsymbol{\tau}_1, \cdots, \boldsymbol{\tau}_T) = \mathcal{L}_t(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x}) - \mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x}). \quad (3)$$

It is obvious that smaller TLD_t suggests that the loss of the merged model is close or even lower than the fine-tuned model on task t, which indicates a better task arithmetic performance.

However, for task arithmetic, it aims to improve the average performance of the final model on all the tasks. Thus, we define the average of all the task loss differences as Average Loss Difference (ALD), which can be formulated as follows:

Definition 2 (Average Task Loss Difference)**.** For the fine-tuned models $\{\boldsymbol{\theta}_i | i = 1, \cdots, T\}$ and task arithmetic merged model $\boldsymbol{\theta}_{\text{final}}$. The average loss difference for all tasks is defined as:

$$\text{ALD}(\lambda_1, \cdots, \lambda_T, \boldsymbol{\tau}_1, \cdots, \boldsymbol{\tau}_T) = \frac{1}{T}\sum_{t=1}^{T}\left(\mathcal{L}_t(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x}) - \mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x})\right). \quad (4)$$

Thus, the optimization objective of MetaGPT is to find the optimal scaling coefficients that can minimize the ALD, which can be formulated as:

Definition 3 (Optimization objective of MetaGPT)**.** Our MetaGPT aims at finding the scaling coefficients $\{\lambda_i | i = 1, \cdots, T\}$, which minimizes the average loss difference ALD:

$$\arg\min_{\lambda_1, \cdots, \lambda_T} \frac{1}{T}\sum_{t=1}^{T}\left(\mathcal{L}_t(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x}) - \mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x})\right). \quad (5)$$

4.3 Separating Data and Coefficients

Before analyzing ALD, we start with reformulating TLD_t by its Taylor expansion.

Lemma 4. *Using Taylor expansion for $\mathcal{L}(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x})$ at $\boldsymbol{\theta}_t$, the TLD_t in Eq. 3 can be reformulated as a quadratic form with respect to the linear combination of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$:*

$$\text{TLD}_t = \frac{1}{2}\boldsymbol{h}_t^\top \left(\int_0^1 \nabla^2 \mathcal{L}_t(\gamma_t(\beta))d\beta\right)\boldsymbol{h}_t, \quad (6)$$

where $\gamma_t(\beta) = \boldsymbol{\theta}_t + \beta(\boldsymbol{\theta}_{\text{final}} - \boldsymbol{\theta}_t)$ and $\boldsymbol{h}_t$ is the linear combination of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$:

$$\boldsymbol{h}_t = \sum_{k \neq t} \lambda_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_0) - (1 - \lambda_t)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0). \quad (7)$$

Single TLD_t is associated with the data, models, and scaling coefficients. As we can see in Eq. 6, we have transformed the data term $\boldsymbol{x}_t$ to the Hessian, the coefficients $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_T]$ and models term $[\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_T]$ to $\boldsymbol{h}$. As our method tends to achieve model-exclusive task arithmetic, the final result should not correlate with the data term. Thus, we first provide a property, which will be used latter in our theorem proofs to separate the data term and scaling coefficients and models term. In general, if a pre-trained network $f(\cdot; \boldsymbol{\theta}_0)$ demonstrates kernel behavior during fine-tuning, i.e., fine-tuning occurs in the linear regime, the following property must be satisfied (Jacot et al., 2018):

Property 5 (NTK linearization)**.** *Around the initialization weights $\boldsymbol{\theta}_0$, a neural network can be approximated with a linear approximation:*

$$f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0)) \approx f(\boldsymbol{x}; \boldsymbol{\theta}_0) + \alpha \cdot C. \quad (8)$$

where $C = (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0)^\top \nabla f(\boldsymbol{x}, \boldsymbol{\theta}_0)$ *is a data and model dependent constant.*

It is worth noting that, as the network width approaches infinity, Eq. 8 becomes exact and remains valid throughout training (Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019), which is specifically suitable for the LLMs arithmetic scenario.

The second property is observed by (Ilharco et al., 2023), which states that the different task vectors are orthogonal:

Property 6 (Orthogonality of Task Vectors)**.** *For task vector* $\boldsymbol{\tau}_i = \boldsymbol{\theta}_i - \boldsymbol{\theta}_0$ *and* $\boldsymbol{\tau}_j = \boldsymbol{\theta}_j - \boldsymbol{\theta}_0$ $(i \neq j)$*, we have the following equation:*

$$\boldsymbol{\tau}_i^\top \boldsymbol{\tau}_j = (\boldsymbol{\theta}_i - \boldsymbol{\theta}_0)^\top (\boldsymbol{\theta}_j - \boldsymbol{\theta}_0) = 0. \tag{9}$$

Now, as we previously introduce our first Lemma to transform the TLD_t in Eq. 3 into a quadratic form with respect to the linear combination of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$. Next, using Property 5,6 and Lemma 7, we can upper bound the TLD_t and separate the data term and scaling coefficients and models term.

Theorem 7. *The* TLD_t *can be upper bounded by:*

$$\begin{aligned}&\mathrm{TLD}_t(\lambda_1, \cdots, \lambda_T, \boldsymbol{\tau}_1, \cdots, \boldsymbol{\tau}_T) \\ &\leq \frac{\delta_t^2}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2^2 \left\{ \sum_{k \neq t}^{T} \mathbb{1}_t(\lambda_k^2) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right\},\end{aligned} \tag{10}$$

where δ_t *is a data-dependent constant and we use* $\mathbb{1}_t(\lambda_k^2)$ *to denote* $(\lambda_k^2)\mathbb{1}(k \neq t) + (1 - \lambda_k^2)\mathbb{1}(k = t)$.

Now, after separating the data-related term to δ_t, the scaling coefficients and models term to $\mathbb{1}_t(\lambda_k^2)$. By summing all the TLD_ts, we can separate the two terms for ALD:

Theorem 8. *By summing all the* TLD_t*, we can separate the correlation between data term and scaling coefficients term in* ALD*:*

$$\begin{aligned}&\mathrm{ALD}(\lambda_1, \cdots, \lambda_T, \boldsymbol{\tau}_1, \cdots, \boldsymbol{\tau}_T) \\ &\leq \sum_{t=1}^{T} \delta_t^2 \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2^2 \left\{ \sum_{k \neq t}^{T} \mathbb{1}(\lambda_k^2) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right\},\end{aligned} \tag{11}$$

4.4 The Optimal Solution

After separating the data term and the scaling coefficients term, we can now reformulate our optimization objective Eq. 11 and derive the closed-form optimal solution of the scaling coefficients.

Theorem 9 (λ decomposition of ALD)**.** *For each* λ_t*, we use it to decompose Eq. 11 as:*

$$\mathrm{ALD} \leq \sum_{t=1}^{T} \mathrm{ALD}_{\lambda_t}, \tag{12}$$

where ALD_{λ_t} *is:*

$$\mathrm{ALD}_{\lambda_t} = \frac{\delta_0^2}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2 \left[\sum_{k=1}^{T} \mathbb{1}_t(\lambda) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right], \tag{13}$$

where $\delta_0 = \max_t \delta_t$. The equation above easily leads to a model-exclusive closed-form solution:

Theorem 10 (Optimal Scaling Coefficients)**.** *We can solve* λ_t *form Eq 13 by:*

$$\lambda_t = \arg\min_{\lambda_t} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2 \left[\sum_{k=1}^{T} \mathbb{1}_t(\lambda) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right]. \tag{14}$$

The above equation is quadratic on λ_t *and the optimal solution for* λ_t *is:*

$$\lambda_t = \frac{\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2}{\sum_{k=1}^{n} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2}. \tag{15}$$

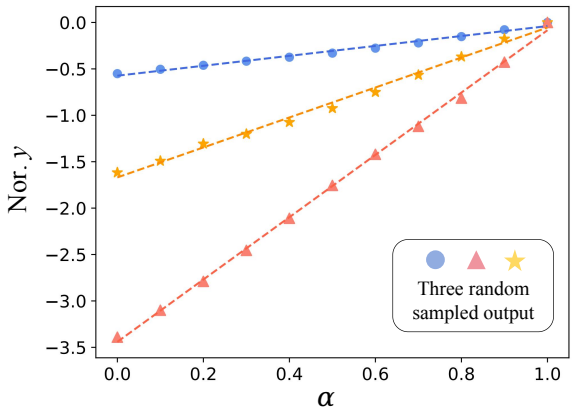

Figure 3: Verification of NTK linearization. We randomly sampled the outputs of Llama-2-7b-chat-hf with different α. We can see that the sampled outputs are linearly with α as expected.

5 Property Verification

In Section 4, we introduced two properties essential to our proof. In this section, we conduct experiments to verify these properties.

5.1 NTK Linearization

Jacot et al. (2018) have proved that when the width of the neural network approaches infinity, it demonstrates kernel behavior and the optimization proceeds in the linear regime. We test Llama-2-7b-chat-hf (Touvron et al., 2023) on AGIEval (Zhong et al., 2023) dataset to verify its

linearity. We have randomly sampled three outputs of the Llama-2-7b-chat-hf when α in Eq. 8 gets value of $[0, 0.1, \cdots, 1]$. For better visualization, we also subtract all the outputs using $\max\{y_i\}$, ensuring they have the same endpoint. From the results in Figure 3, we can see that all the outputs are almost linear with α, which indicates that LLMs do exhibit a kernel behavior during finetuning.

5.2 Task Vector Orthogonality

Ilharco et al. (2023); Yang et al. (2023b) have performed experiments to verify this property for vision models. For LLMs, we also observe similar results: these task vectors are almost orthogonal to each other. The result has been shown in Figure 4. We can see that different task vectors are almost orthogonal, and their cosine similarity is nearly 0 as Eq.9 expected, which verifies the property we have used for our proof.

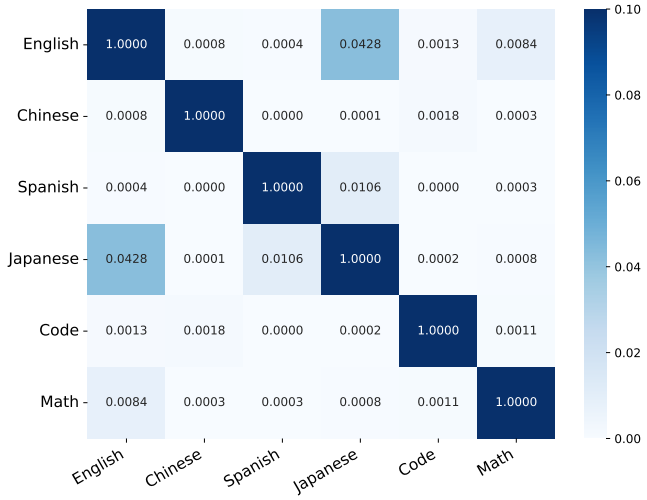

Figure 4: Verification of orthogonality. We calculate the cosine similarity between six different task vectors and find that their cosine similarity is nearly 0.

6 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our `MetaGPT`. In the first section, we demonstrate that our `MetaGPT` consistently achieves optimal average performance across diverse datasets and is robust for model series with varying parameter sizes and architectures. DARE and Ties-Merging are task vector-improving methods that resolve conflicts and redundant parameters between task vectors. We conduct experiments to demonstrate that our method is orthogonal to theirs and can be integrated to improve the average performance further. Finally, we show that the model merged by our `MetaGPT` has better out-of-distribution generalization ability.

6.1 Merging Models Using `MetaGPT`

Dataset and Models. To test the effectiveness of our method, we use Llama-2-7b-chat-hf (Touvron et al., 2023), MAmmoTH-7B (Yue et al., 2023) and llama-2-coder-7b (Manuel Romero, 2023) as models fine-tuned on general knowledge, math, and code datasets using the pre-trained model Llama-2-7B-hf (Touvron et al., 2023). Moreover, we use a different model architecture: Mistral-7B-Instruct-v0.2 (AI), MAmmoTH2-7B-Plus (Yue et al., 2024) and Mistral-7B-codealpaca-lora (Nondzu) as models fine-tuned on general knowledge, math, and code datasets using pre-trained model Mistral 7B (Jiang et al., 2023). We also provide experiments using models with larger sizes: Llama-2-13b-chat-hf (Touvron et al., 2023), MAmmoTH-13B (Yue et al., 2023), and llama-2-13b-code-chat (TAŞAR, 2023) as models fine-tuned on general knowledge, math, and code datasets using the pre-trained model Llama-2-13B-hf (Touvron et al., 2023). We use WinoGrande (Sakaguchi et al., 2021) and AGIEval (Zhong et al., 2023) for evaluating general knowledge performance, GSM8K (Cobbe et al., 2021) and MATH (Saxton and Hill, 2019) for testing mathematical reasoning ability, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for estimating code-generation capacity.

Evaluation Metrics. We use common evaluation settings for a single task: 5-shot accuracy for AGIEval, 4-shot accuracy for GSM8K and MATH, 3-shot accuracy for MBPP, and zero-shot accuracy for HumanEval and WinoGrande. We employ two key metrics in evaluating different merging methods: absolute average performance and normalized average accuracy.

Quantitative Evaluation for LLaMA-2-7B. We use the metrics and datasets we introduced above to evaluate the performance of different methods. We use Weight Average (Wortsman et al., 2022), Task Arithmetic (Ilharco et al., 2023), Ties-Merging (Yadav et al., 2024) and DARE (Yu et al., 2023), which are also model exclusive and computationally efficient methods, to compare with our method by merging LLaMA-2-7B. The scores in Table 1 show that for WinoGrande, AGIEval, GSM8k, and MATH dataset, our method scores 64.25, 32.71,

Table 1: Performance comparison of merging different LLaMA-2-7B fine-tuned models on different datasets.

Model	WinoGrande	AGIEval	GSM8k	MATH	MBPP	HumanEval	Abs. Avg	Nor. Avg
LM	62.67	34.01	28.66	4.00	22.00	7.31	26.44	0.91
Math	61.64	29.40	47.16	2.40	17.40	11.58	28.26	0.84
Code	61.88	27.41	17.21	2.20	24.80	21.92	25.90	0.84
Weight Average	63.93	31.36	37.68	7.00	23.40	**20.12**	30.58	1.25
Task Arithmetic	63.54	31.70	37.53	5.20	23.20	19.51	30.11	1.12
Ties Merging	62.67	32.10	37.93	7.40	22.80	18.29	30.20	1.26
DARE	63.27	32.25	37.86	7.00	**24.40**	19.51	30.72	1.26
MetaGPT(ours)	**64.25**	**32.71**	**45.41**	**7.80**	21.20	17.68	**31.51**	**1.31**

Table 2: Performance comparison of merging different Mistral-7B fine-tuned models on different datasets.

Model	WinoGrande	AGIEval	GSM8k	MATH	MBPP	HumanEval	Abs. Avg	Nor. Avg
LM	69.30	37.55	47.54	7.80	34.40	34.75	38.56	0.776
Math	63.46	38.06	68.46	28.00	24.00	25.00	41.16	0.854
Code	67.32	40.69	60.73	15.60	43.40	39.02	44.46	0.917
Weight Average	67.88	41.12	62.77	17.40	**40.20**	38.41	44.63	0.921
Task Arithmetic	67.88	41.41	63.38	18.80	**40.20**	38.40	45.01	0.932
Ties Merging	67.72	41.06	60.35	17.80	**40.20**	40.24	44.56	0.924
DARE	67.40	40.58	59.67	19.00	36.00	**40.85**	43.92	0.913
MetaGPT(ours)	**68.35**	**41.86**	**66.03**	**20.80**	39.00	35.37	**45.24**	**0.936**

45.41, and 7.80, which outperforms other methods. For the HumanEval dataset, DARE performs best, and for the MBPP dataset, the Weight Average method achieves the highest score. Since our method aims to achieve the *average best performance*, we use absolute average performance score and normalized average performance score to compare the five methods. We can see that our MetaGPT achieves the rank-1 score 31.51, 1.31 in both absolute average performance and normalized average performance.

Using Different Model Architecture. We also use a different model architecture, Mistral-7B, for evaluation, and the result has been shown in Table 2. The scores in Table 2 show similar results to LLaMA-2-7B: For WinoGrande, AGIEval, GSM8k, and MATH dataset, our MetaGPT scores 41.86, 68.35, 66.03, 20.8, which outperforms existing methods, for HumanEval dataset Weight Average, Task Arithmetic, and Ties Merging performs best and for MBPP dataset, DARE method achieves the highest score.

Using Larger Model Size. We also test our method using a larger model LLaMA-2-13B (Touvron et al., 2023). The scores in Table 3 demonstrate that for AGIEval, Math, and MBPP datasets, our method outperforms other methods. For WinoGrand, GSM8K, and HumanEval dataset, DARE, Weight Average and Ties-Merging achieves the highest score. Similarly, under the *average measure* absolute average performance and normalized average performance, our method also outperforms the other five methods.

Integrate with Ties/DARE As there are conflicts and redundant parameters between task vectors, DARE (Yu et al., 2023) and Ties-Merging (Yadav et al., 2024) are two methods trying to solve the interfaces, reducing the redundancy and thereby improving the performance of task arithmetic. Since our method is also based on the framework of task arithmetic, Ties-merging and DARE are expected to improve the performance of our MetaGPT further. As we can see in Table 4, under the baseline of Ties-Merging and DARE methods, our method is orthogonal to Ties-Merging and DARE and can integrate them into our MetaGPT, thus leading to

Table 3: Comparison of performance of merging fine-tuned LLaMA-2-13B on different datasets.

Model	WinoGrande	AGIEval	GSM8K	MATH	MBPP	HumanEval	Abs. Avg	Nor. Avg
LM	64.80	35.04	42.84	4.80	27.00	15.24	31.62	1.02
Math	60.38	36.74	55.27	3.40	22.60	12.80	31.87	0.93
Code	63.93	32.04	36.47	5.00	26.60	16.46	30.08	1.01
Weight Average	64.88	37.23	**53.15**	7.60	29.80	21.95	35.77	1.29
Task Arithmetic	65.11	35.48	50.34	7.20	29.80	21.95	34.98	1.25
Ties Merging	65.23	36.02	51.23	7.40	30.20	**23.17**	35.54	1.28
DAREs	**65.70**	36.87	51.85	7.60	30.00	22.56	35.76	1.29
MetaGPT(ours)	65.04	**37.33**	52.92	**7.80**	**30.40**	21.95	**35.91**	**1.30**

Table 4: MetaGPT can be integrated with DARE and Ties-Merging, thereby leading to further improvment.

Method	WinoGrande	AGIEval	GSM8k	MATH	MBPP	HumanEval	Abs. Avg	Nor. Avg
Ties-Merging	62.67	32.10	37.93	7.40	22.80	18.29	30.20	1.26
Ties + MetaGPT	62.35	32.91	46.10	8.00	22.40	17.68	**31.57**	**1.33**
Dare	63.27	32.25	37.86	7.00	24.40	19.51	30.72	1.26
Dare + MetaGPT	62.99	33.01	45.72	7.60	21.80	18.29	**31.57**	**1.30**

further improvement. For example, the average absolute performance of DARE has been improved by our MetaGPT from 30.72 to 31.57. And the normalized absolute performance of DARE has been improved by our MetaGPT from 1.26 to 1.3. Ties-merging also leads to a similar conclusion: the average absolute performance of DARE has been improved by our MetaGPT from 30.20 to 31.57. And the normalized absolute performance of DARE has been improved by our MetaGPT from 1.26 to 1.33.

6.2 Out of Distribution Generalization

Following (Yang et al., 2023b; Jin et al., 2022), we also compare the out-of-distribution generalization ability of different merging methods. We evaluate different methods using JEC-QA (Zhong et al., 2020), FinanceIQ (DI, 2023), and MedQA (Jin et al., 2021) dataset. All three datasets use 5-shot accuracy as the evaluation metric. Table 5 summarizes out-of-distribution generalization performance when merging all domain specific models using different methods. As we can see, MetaGPT outperforms current methods on these unseen datasets, which demonstrates that MetaGPT is more robust to the test data distribution shifts.

Table 5: Out of distribution Generalization

Model	JEC-QA	FinancelQ	MedQA	Avg
LM	31.32	32.83	30.20	31.45
Math	25.56	30.25	24.73	26.85
Code	29.23	30.87	26.25	28.78
Weight Average	30.73	34.17	29.90	31.60
Task Arithmetic	30.85	33.89	30.13	31.62
Ties Merging	30.80	33.53	30.02	31.45
DARE	30.79	33.93	**30.17**	31.63
MetaGPT(ours)	**30.97**	**34.31**	30.07	**31.78**

7 Conclusion

In this paper, we have provided a novel model merging method named MetaGPT, an efficient and optimal model-exclusive task arithmetic specifically designed for LLMs. We provide the mathematical formulation of task arithmetic's optimization objective and the theoretical analysis of the task arithmetic performance bound. By separating the data and scaling coefficient term under careful approximation, the closed-form solution provides an avenue for optimally achieving task arithmetic without using any data. Extensive experiment results show that our MetaGPT outperforms the existing state-of-the-art model-exclusive merging method and can be integrated with task vector-improving methods such as Ties-Merging and DARE.

8 Limitations

(1) Our works share the same general limitation of existing task arithmetic based methods: Our merging method relies on common initialization and model architecture, which ensures that the task vectors are orthogonal. (2) Moreover, since our method is specifically designed for LLMs and relies on the NTK linearization, for small size models, our method may not perform well.

References

Mistral AI. Mistral-7b-instruct-v0.2.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. 2019. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. 2022. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *Advances in Neural Information Processing Systems*, 35:8265–8277.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. 2021. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

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

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *NeurIPS*.

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. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Duxiaoman DI. 2023. Financeiq.

Ke Ding, Xin Dong, Yong He, Lei Cheng, Chilin Fu, Zhaoxin Huan, Hai Li, Tan Yan, Liang Zhang, Xiaolu Zhang, et al. 2021. Mssm: a multiple-level sparse sharing model for efficient multi-task learning. In *SIGIR*, pages 2237–2241.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. 2021. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516.

Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. 2020. Learning to branch for multi-task learning. In *ICML*, pages 3854–3863. PMLR.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. 2021. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *NeurIPS*, 34:29335–29347.

Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo, and James Caverlee. 2022. Metabalance: Improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks. *WWW*, pages 2205–2215.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. *The Twelfth International Conference on Learning Representations*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

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*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. 2019. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. *NeurIPS*, 34:18878–18890.

Shikun Liu, Edward Johns, and Andrew J. Davison. 2019. End-to-end multi-task learning with attention. In *CVPR*, pages 1871–1880. Computer Vision Foundation / IEEE.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, pages 5334–5343.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *SIGKDD*, pages 1930–1939. ACM.

Manuel Romero. 2023. llama-2-coder-7b (revision d30d193).

Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *CVPR*, pages 3994–4003. IEEE Computer Society.

Nondzu. Mistral-7b-codealpaca-lora.

OpenAI. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2024. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. 2023. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 28656–28679. PMLR.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Grefenstette Saxton and Kohli Hill. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv:1904.01557*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *NeurIPS*, pages 525–536.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. 2020. Adashare: Learning what to share for efficient deep multi-task learning. *NeurIPS*, 33:8728–8740.

Davut Emre TAŞAR. 2023. llama-2-13b-code-chat.

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*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Enneng Yang, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo. 2023a. Adatask: A task-aware adaptive learning rate approach to multi-task learning. In *AAAI*, volume 37, pages 10745–10753.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023b. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *NeurIPS*, 33:5824–5836.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*.

Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. Jec-qa: a legal-domain question answering dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9701–9708.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Appendix

A Proof

A.1 Proof of Lemma 4

Using Taylor expansion for $\mathcal{L}(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x})$ at $\boldsymbol{\theta}_0$:

$$\mathcal{L}(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x}) \tag{16}$$

$$= \mathcal{L}_t\Big(\sum_{k=1}^{n} \lambda_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_0) + \boldsymbol{\theta}_0, \boldsymbol{x}_t\Big) \tag{17}$$

$$= \mathcal{L}_t(\boldsymbol{h}_t + \boldsymbol{\theta}_t, \boldsymbol{x}_t) \tag{18}$$

$$= \mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x}_t) + \nabla\mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x}_t)\boldsymbol{h}_t + \frac{1}{2}\boldsymbol{h}_t^\top \left(\int_0^1 \nabla^2 \mathcal{L}_t(\gamma_t(\beta))d\beta\right)\boldsymbol{h}_t \tag{19}$$

where $\gamma_t(\beta) = \boldsymbol{\theta}_t + \beta(\boldsymbol{\theta}_{\text{final}} - \boldsymbol{\theta}_t)$ and $\boldsymbol{h}_t$ is the linear combination of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$:

$$\boldsymbol{h}_t = \sum_{k\neq t} \lambda_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_0) - (1-\lambda_t)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) \tag{20}$$

Because the $\boldsymbol{\theta}_t$ is fine-tuned using loss $\mathcal{L}_t$, the gradient of $\mathcal{L}_t$ at $\boldsymbol{\theta}_t$ is zero, and the first order expansion is 0. Substituting Eq. 19 to Eq. 3, we have:

$$\text{TLD}_t = \mathcal{L}_t(\boldsymbol{\theta}_{\text{final}}, \boldsymbol{x}_t) - \mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x}_t) \tag{21}$$

$$= \frac{1}{2}\boldsymbol{h}_t^\top \left(\int_0^1 \nabla^2 \mathcal{L}_t(\gamma_t(\beta))d\beta\right)\boldsymbol{h}_t \tag{22}$$

Thus, we have completed the proof.

A.2 Proof of Theorem 7

Before starting the proof, we first introduce a lemma:

Lemma 11. *Under the Property. 5, the task vector is linearly with the gradient.*

$$\delta_t(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) = \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \tag{23}$$

Proof: For gradient descent, we have:

$$\boldsymbol{\theta}_t - \boldsymbol{\theta}_0 = \sum_{i=1}^{n} lr_i \nabla \mathcal{L}_t^i \tag{24}$$

$$= \sum_{i=1}^{n} lr_i \frac{\partial \mathcal{L}_t^i}{\partial f}\nabla f_i \tag{25}$$

where lr_i and $\nabla\mathcal{L}_t^i$ and ∇f_i is the learning rate, gradient loss, gradient of f at step i. From Property 5, we can see that the fine-tuning process of f occurs in the linear regime, which indicates that the first order derivative in the task vector direction is an constant. We derivative at $\boldsymbol{\theta}_t$:

$$\nabla_{\boldsymbol{\theta}_t} f(\boldsymbol{x}, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \tag{26}$$

Thus, we substitute all the gradient of f_i using $\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{x}, \boldsymbol{\theta}_0)$:

$$\delta_t(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) = \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \tag{27}$$

$$\text{where} \quad \frac{1}{\delta_t} = \sum_{i=1}^{n} lr_i \frac{\partial \mathcal{L}_t^i}{\partial f}$$

. Thus, we have completed the proof of the Lemma.

For the of loss function, using Property 5 we have:

$$\mathcal{L}_t(\boldsymbol{\theta}_t, \boldsymbol{x}_t) = \|f(\boldsymbol{x}_t, \boldsymbol{\theta}_t) - y\|^2 \tag{28}$$

$$= \|(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0)^\top \nabla f(\boldsymbol{x}_t; \boldsymbol{\theta}_0) + C_0\|^2 \tag{29}$$

For the Hessian of loss function, it can be represented as:

$$\nabla^2_{\boldsymbol{\theta}_t} \mathcal{L}_t = \nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{x}_t; \boldsymbol{\theta}_0) \nabla^\top_{\boldsymbol{\theta}_0} f(\boldsymbol{x}_t; \boldsymbol{\theta}_0) \tag{30}$$

Using Eq. 30 the TLD_t can be represented as:

$$2\text{TLD}_t \tag{31}$$

$$= \boldsymbol{h}_t^\top \left(\int_0^1 \nabla^2 \mathcal{L}_t(\gamma_t(\beta))d\beta\right)\boldsymbol{h}_t \tag{32}$$

$$= \boldsymbol{h}_t^\top \left(\nabla^2 \mathcal{L}_t(\tilde{\boldsymbol{\theta}})\right)\boldsymbol{h}_t \tag{33}$$

$$= \boldsymbol{h}_t^\top \left(\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\nabla_{\boldsymbol{\theta}_0} f^\top(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\right)\boldsymbol{h}_t \tag{34}$$

$$= \text{tr}\left\{\boldsymbol{h}_t^\top \left(\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\nabla_{\boldsymbol{\theta}_0} f^\top(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\right)\boldsymbol{h}_t\right\} \tag{35}$$

$$\leq \text{tr}(\boldsymbol{h}\boldsymbol{h}^\top)\text{tr}\left(\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\nabla_{\boldsymbol{\theta}_0} f^\top(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\right) \tag{36}$$

For $\text{tr}(\boldsymbol{h}\boldsymbol{h}^\top)$, using Property. 6, we have:

$$\text{tr}(\boldsymbol{h}\boldsymbol{h}^\top) \tag{37}$$

$$= \left\|\sum_{k\neq t}^{T} \lambda_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_0) - (1-\lambda_t)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0)^\top\right\|^2 \tag{38}$$

$$= \sum_{k\neq t}^{T} \left[\mathbb{1}_{k\neq t}(\lambda_k^2) + \mathbb{1}_{k=t}(1-\lambda_k^2)\right]\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \tag{39}$$

$$= \sum_{k\neq t}^{T} \left[\mathbb{1}(\lambda_k^2)\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2\right] \tag{40}$$

where $(\lambda_k^2)\mathbb{1}(k \neq t) + (1-\lambda_k^2)\mathbb{1}(k = t) := \mathbb{1}_t(\lambda_k^2)$.

For the second part: $\mathrm{tr}(\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0, \boldsymbol{x}_t) \nabla_{\boldsymbol{\theta}_0} f^\top(\boldsymbol{\theta}_0, \boldsymbol{x}_t))$, using Lemma 11 we can have:

$$\mathrm{tr}\left(\nabla_{\boldsymbol{\theta}_0} f(\boldsymbol{\theta}_0, \boldsymbol{x}_t) \nabla_{\boldsymbol{\theta}_0} f^\top(\boldsymbol{\theta}_0, \boldsymbol{x}_t)\right) = \delta_t^2 \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2 \tag{41}$$

Thus, for TLD_t we can upper bound it by

$$\mathrm{TLD}_t \leq \frac{\delta_t^2}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2^2 \left\{ \sum_{k \neq t}^{T} \mathbb{1}_t(\lambda_k^2) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right\} \tag{42}$$

A.3 Proof of Theorem 8

By summing Eq.42 from 1 to T, we can complete the proof.

$$\mathrm{ALD} \leq \sum_{t=1}^{T} \frac{\delta_t^2}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2^2 \left\{ \sum_{k \neq t}^{T} \mathbb{1}_t(\lambda_k^2) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right\} \tag{43}$$

A.4 Proof of Theorem 9

First, for Eq. 43, we have:

$$\mathrm{ALD} \leq \frac{\delta_t^2}{2} \sum_{t=1}^{T} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2^2 \left\{ \sum_{k \neq t}^{T} \mathbb{1}_t(\lambda_k^2) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right\} \tag{44}$$

where $\delta_0 = \max\{\delta_i\}$ For Eq. 44, it is easy to verify that the terms containing λ_t can be represented as:

$$\mathrm{ALD}_{\lambda_t} = \frac{\delta_t^2}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2 \left[\sum_{k=1}^{T} \mathbb{1}_t(\lambda) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right] \tag{45}$$

Thus, the ALD can be upper bounded by

$$\mathrm{ALD} \leq \sum_{t=1}^{T} \mathrm{ALD}_{\lambda_t} \tag{46}$$

A.5 Proof of Theorem 10

Because each ALD_{λ_t} does not contain other scaling coefficients. We can solve each optimal λ_t from ALD_{λ_t}:

$$\lambda_t = \arg\min_{\lambda_t} \frac{\delta_0^2}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2 \left[\sum_{k=1}^{T} \mathbb{1}_t(\lambda) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right] \tag{47}$$

$$= \arg\min_{\lambda_t} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2 \left[\sum_{k=1}^{T} \mathbb{1}_t(\lambda) \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2 \right] \tag{48}$$

The RHS of the above equation is quadratic on λ_t and and the optimal solution for λ_t is:

$$\lambda_t = \frac{\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|^2}{\sum_{k=1}^{n} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}_0\|^2} \tag{49}$$

B Details of Models and Datasets

Table 7 shows the versions and correspondence with pre-trained backbones of fine-tuned LLMs. Table 8 shows the details of the datasets we use in our paper.

C Infra and hardware details

We use PyTorch as the deep learning framework. We merge and evaluate the neural networks using A100 GPUs.

D Hyper-parameter Setting

For both DARE and TIES-Merging, the density of 0.55 is used, and the open-source tool MergeKit[1] is employed for the merging process.

E Details of different Methods

We give a detailed comparison of the current merging method below from the perspective of extra data information, time complexity, and optimal performance. The time complexity for forward and backward processes is denoted as FW and BP. For RegMean, it requires the inner product data matrices for layer input to calculate the updated parameters. It only requires a forward process, but loading all the inner products of the layer input matrix requires $O(\theta^2)$ memory. For Fisher merge, it also requires the data to calculate the Fisher Matrix, which requires the forward process to calculate the Fisher matrix and $O(\theta^2)$ memory to store the Fisher matrix. Grid-search Task Arithmetic (G-Task Arithmetic) requires $O(G^{\mathrm{T}} \times \mathcal{T}_{\mathrm{FW}})$ forward process to evaluate, where G is the grid number (G = 100 means 100 girds from 0 to 1) and T is the number of tasks. The space complexity is also equal to the memory requirement of the forward process. For Adamerging, it simultaneously loads T LLMs to optimize, whose time complexity is $O(\mathcal{T}_{\mathrm{BP}})$ and space complexity is: $O(\mathcal{S}_{\mathrm{BP}} \times T)$. For weight average, task arithmetic, and `MetaGPT`, they all do not need extra data information, which is model exclusive. Their time and space complexity is $O(1)$ and $O(n)$, but only our `MetaGPT` achieves optimal performance.

[1] https://github.com/arcee-ai/mergekit/tree/main

Table 6: Extra data information requirement, time and space complexity, and optimally of current methods. The time complexity for forward process and back propagation are denote by $\mathcal{T}_{\text{FW}}, \mathcal{T}_{\text{BP}}$. The space complexity for forward process and back propagation are denote by $\mathcal{S}_{\text{FW}}, \mathcal{S}_{\text{BP}}$. T is the number of task, θ is the number of parameters and G is the grid number (G = 100 means 100 girds from 0 to 1).

	Extra Data Info	Time Complexity	Space Complexity	Optimal	Apply to LLMs
RegMean	✓	$O(\mathcal{T}_{\text{FW}})$	$O(\theta^2)$	✓	✗
Fisher Merge	✓	$O(\mathcal{T}_{\text{FW}})$	$O(\theta^2)$	✓	✗
G-Task Arithmetic	✓	$O(G^{\text{T}} \times \mathcal{T}_{\text{FW}})$	$O(\mathcal{S}_{\text{FW}})$	✓	✗
AdaMerging	✓	$O(\mathcal{T}_{\text{BP}})$	$O(\mathcal{S}_{\text{BP}} \times T)$	✓	✗
Task Arithmetic	✗	$O(1)$	$O(n)$	✗	✓
Weight Average	✗	$O(1)$	$O(n)$	✗	✓
MetaGPT	✗	$O(1)$	$O(n)$	✓	✓

Table 7: Details of datasets we used for our evaluation.

Dataset	Number of Training Examples	Number of Validation Examples	Number of Testing Examples	Evaluate Metric
WinoGrande	9248	1267	1767	0-shot accuracy
AGIEval	N/A	N/A	8062	5-shot accuracy
GSM8k	7473	N/A	1319	4-shot accuracy
Math	7500	N/A	1500	4-shot accuracy
MBPP	374	30	500	3-shot accuracy
HumanEval	N/A	N/A	164	0-shot accuracy
JEC-QA	N/A	N/A	26365	5-shot accuracy
FinancelQ	N/A	N/A	7173	5-shot accuracy
MedQA	N/A	N/A	61097	5-shot accuracy

Table 8: Details of models we used for our evaluation.

Pre-trained Model	Task	Fine-tuned-Models
LLaMA-2-7b	General Knowledge	meta-llama/Llama-2-7b-chat-hf
	Mathematical Reasoning	TIGER-Lab/MAmmoTH-7B
	Code Generating	mrm8488/llama-2-coder-7b
	Chinese	hfl/chinese-llama-2-7b
	Spanish	clibrain/Llama-2-7b-ft-instruct-es
	Japanese	elyza/ELYZA-japanese-Llama-2-7b
Mistral-7b	General Knowledge	mistralai/Mistral-7B-Instruct-v0.2
	Mathematical Reasoning	TIGER-Lab/MAmmoTH2-7B
	Code Generating	Nondzu/Mistral-7B-codealpaca-lora
LLaMA-2-13b	General Knowledge	meta-llama/Llama-2-13b-chat-hf
	Mathematical Reasoning	TIGER-Lab/MAmmoTH-13B
	Code Generating	emre/llama-2-13b-code-chat