Enhancing LLM Fine-Tuning via Selective Parameter Merging

Anonymous ACL submission

Abstract

Supervised fine-tuning (SFT) is a crucial technique for tailoring the generalization capacity of Large Language Models (LLMs) to specific target tasks. This study investigates enhancing LLMs fine-tuning through the parameter merging technique. By merging models fine-tuned with varied data order, we achieve an enhanced SFT model demonstrating improved performance and lower validation losses. To our best knowledge, this is the first introduction of "parameter-selection merging" technique, which innovatively merges models by selecting parameters from one sub-model in each parameter dimension, surpassing traditional weighted-average method across 5 datasets. Furthermore, this method has also shown superiority in multi-task merging scenarios, indicating a promising avenue for future LLM optimizations.

1 Introduction

Thanks to the substantial expansion of training scale and model size, large language models (LLMs) have achieved significant breakthroughs across a broad spectrum of NLP tasks (Radford et al., 2019; Touvron et al., 2023). For downstream tasks, supervised fine-tuning (SFT) is a crucial technique for LLMs, enabling the customization of pre-trained models for specialized tasks and domains (Dettmers et al., 2023; Zhao et al., 2023).

Parameter merging, defined as combining multiple models within the parameter space (Matena and Raffel, 2022), primarily focuses on integrating SFT models for different tasks into one capable of addressing all associated sub-tasks. Numerous related studies have been conducted in this field. For example, Wortsman et al. (2022) and Jin et al. (2022) employed linear matrix transformation for task adaptability; Yadav et al. (2023) addressed the issue of sign conflicts across different sub-tasks; Similarly, Yu et al. (2023a) mitigated task conflict by partially removing task-specific parameters; Moreover, Xiao et al. (2023) aimed to maximally preserve the performance of one primary task among all tasks; Furthermore, Huang et al. (2023) investigated the composability of LoRA (Hu et al., 2021) for enhancing cross-task generalization.

However, compared to merging models from multiple tasks, which often leads to performance degradation on individual tasks, the potential of utilizing the parameter merging technique to enhance single-task LLMs has not yet received much attention. While some studies, such as Wortsman et al. (2022), have explored merging models fine-tuned with different settings, these experiments were predominantly conducted on comparatively smaller models like BERT (Kenton and Toutanova, 2019) and achieved only modest improvements. To bridge the gap, this work investigates enhancing LLM fine-tuning through the parameter merging technique. Our experiments span various models, including BERT, TinyLlama, Llama2, and WizardModel (Kenton and Toutanova, 2019; Zhang et al., 2024; Touvron et al., 2023; Luo et al., 2023; Xu et al., 2023), covering tasks such as SST-2, SQuAD, Alpaca, GSM8K, among others (Socher et al., 2013; Rajpurkar et al., 2016; Taori et al., 2023; Cobbe et al., 2021). The core contributions of this paper are summarized as follows:

- We discover that merging sub-models fine-tuned with different data orders can yield an enhanced LLM for the target task, achieving better performance and lower validation loss. We also observe that the average performance gains for Llama models surpass those for BERT models, suggesting the potential of this method in large model contexts.

- Introducing "**parameter-selection merging**" technique for the first time, which innovatively merges models by selecting parameters from one sub-model in each parameter dimension.

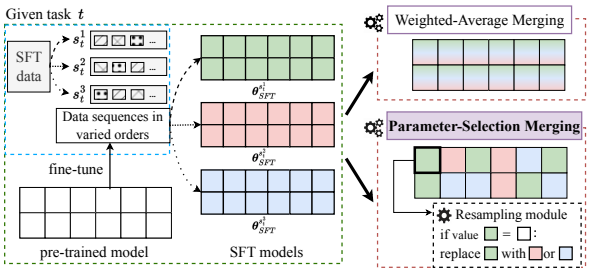

Figure 1: Illustration comparing two parameter merging techniques. Weighted-average merging calculates the weighted sum of all sub-model parameters at each parameter dimension, whereas parameter-selection merging selects parameters from a single sub-model. In the resampling module, parameters from the chosen sub-model that equals those of the pre-trained model are replaced with parameters from alternative ones.

This approach not only outperforms the traditional weighted-average merging method but also can be further enhanced with a straightforward resampling strategy.

- Finally, we conduct experiments showcasing the effectiveness of parameter-selection merging in multi-task fusion, demonstrating its general applicability across various scenarios.

2 Method

2.1 Merge Fine-tuned LLMs with Different Data Order

In this work, we enhance LLM fine-tuning by merging models fine-tuned with different data orders. As depicted in Figure 1, for a given task t, the method initiates by fine-tuning a pre-trained LLM multiple times, each with a uniquely ordered data sequence. Specifically, for various data sequences $\{s_t^1, s_t^2, \cdots, s_t^k\}$, we obtain a set of SFT models $\{\boldsymbol{\theta}_{SFT}^{s_t^1}, \boldsymbol{\theta}_{SFT}^{s_t^2}, \cdots, \boldsymbol{\theta}_{SFT}^{s_t^k}\}$. Subsequently, these variously fine-tuned models are integrated into a unified model through parameter merging techniques, yielding the enhanced model $\boldsymbol{\theta}_{\text{SFT}}\uparrow$

2.2 Parameter-Selection Merging

Existing parameter merging techniques can generally be categorized under "weighted-average merging" approach. In this work, we introduce a novel parameter merging approach: "**parameter-selection merging**." Figure 1 shows the comparison of two merging techniques: weighted-average merging calculates the weighted sum of all sub-model parameters at each parameter dimension. Given a set of K sub-models $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K\}$, with each model $\boldsymbol{\theta}_i$ consisting of parameters $\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,d}$ for parameter dimension d, weighted-average merging can be represented by the formula:

$$\theta_{\text{merged},j} = \sum_{i=1}^{K} w_i \theta_{\text{i,j}}, \quad \forall j \in \{1, \ldots, d\} \qquad (1)$$

where $\theta_{i,j}$ is the parameter of the i-th sub-model in dimension d, w_i is the weight applied to $\theta_{i,j}$.

Conversely, parameter-selection merging selects a parameter from a single sub-model for each dimension with probbability p_i, represented by the formula:

$$\theta_{\text{merged},j} = \theta_{i,j} \text{ with } p_i, \quad \forall j \in \{1, \ldots, d\} \qquad (2)$$

where p_i is the probability $\theta_{i,j}$ is selected. Given that each sub-model in our method is fine-tuned on the same training dataset, we assign equal weight w_i and selection probability p_i to each sub-model: $w_i = \frac{1}{K}, \quad p_i = \frac{1}{K}$, where K is the total number of sub-models.[1]

2.3 Resample Strategy

Task Vectors. Let $\boldsymbol{\theta}_{\text{pre}}$ represent the pre-trained model's weights and $\boldsymbol{\theta}_{\text{SFT}}$ denote weights after supervised fine-tuning for task t. The task vector τ is defined to capture task-specific adaptations, calculated as: $\boldsymbol{\tau} = \boldsymbol{\theta}_{\text{SFT}} - \boldsymbol{\theta}_{\text{pre}}$ (Ilharco et al., 2022).

Guided by the intuition to maximize the impact of task vectors, we introduce a resampling method within the parameter-selection merging framework to further improve the merged model performance. The task vector $\tau_{i,j}$ represents task vector of the i-th sub-model at parameter dimension j. As depicted in Figure 1, if $\tau_{i,j} = 0$, indicating no parameter change after fine-tuning, a new parameter is resampled from the pool of all sub-models.[2] This procedure can be iterated n times, where n is a predefined hyperparameter, as formalized below:

$$\theta_{\text{merged},j}^{(n)} = \begin{cases} \theta_{i,j} & \text{if } \tau_{i,j} \neq 0 \text{ or } n = 0, \\ \theta_{\text{merged},j}^{(n-1)} & \text{others,} \end{cases} \qquad (3)$$

Specifically, $\theta_{\text{merged},j}^{0}$ equals parameter-selection merging without resampling module.

[1] Although Wortsman et al. (2022) explores assigning w_i based on sub-model performance (Greedy Soup), we use equal weights due to performance disparities arising from training data order is negligible.

[2] This strategy allows for parallel operations on tensors by considering all sub-models during resampling.

Model	Method	SST-2 acc	MNLI acc	SQuAD EM	Avg Δ
BERT-base	avg / best	91.93 / **92.66**	83.99 / 84.15	81.07 / 81.36	+ 0.54 / 0.15
	merged	92.09(+0.16/-0.57)	**84.28**(+0.29/+0.13)	**82.25**(+1.18/+0.89)	
BERT-large	avg / best	93.44 / 93.92	86.42 / **86.65**	84.15 / 84.56	+ 0.60 / 0.15
	merged	**94.04**(+0.60/-0.12)	86.38(-0.04/-0.27)	**85.39**(+1.24/+0.83)	
TinyLlama	avg / best	94.81 / 95.64	85.46 / 85.81	80.53 / 81.46	+ 1.42 / 0.72
	merged	**95.76**(+0.95/+0.12)	**86.64**(+1.18/+0.83)	**82.66**(+2.13/+1.20)	
Llama-2-7b	avg / best	95.09 / 96.56	88.84 / 89.28	84.53 / 85.31	+ 1.86 / 0.96
	merged	**96.79**(+1.70/0.23)	**90.37**(+1.53/+1.09)	**86.87**(+2.34/+1.56)	

Table 1: Performance comparison between single and merged SFT models across various pre-trained models. The terms “avg" and “best" denote the average performance across all sub-models and the highest performance observed among them, respectively.

Method	AlpacaEval win-rate	GSM8K acc	GSM8K-RFT acc	MATH acc	HumanEval pass@1	Avg Δ
single SFT	24.25	41.29	52.74	10.36	26.82	-
weighted-avg	24.97(+0.72)	44.35(+3.06)	53.29(+0.88)	11.24(+0.55)	26.22(-0.60)	+ 0.92
param-selection	25.66(+1.41)	44.73(+3.44)	53.35(+0.61)	11.37(+1.01)	27.43(+0.61)	+ 1.42
. + resample	**25.91**(+1.66)	**45.26**(+3.97)	**54.32**(+1.58)	**12.00**(+1.64)	**28.05**(+1.23)	**+ 2.02**

Table 2: Performance comparison of weighted-average and parameter-selection merging based on Llama-2-7b. “weighted-avg" means weighted-average and “param-selection" means parameter-selection merging method.

3 Experiments

3.1 Datasets and Evaluation Metrics

Datasets. Datasets employed in our experiments for fine-tuning include 3 traditional tasks: SST-2 (Xu et al., 2023) (sentiment classification), MNLI (Williams et al., 2017) (natural language inference), and SQuAD (Rajpurkar et al., 2016) (question answering); and 5 tasks desgined for LLMs: Stanford Alpaca (Taori et al., 2023) (instruction-following), GSM8K (Cobbe et al., 2021), GSM8K-RFT (Yuan et al., 2023) and MATH (Hendrycks et al., 2021) (mathematical reasoning), and Evol-instruction-66k (code generating)[3].

Evaluation Metrics. We use AlpacaEval (Li et al., 2023) to evaluate models fine-tuned on Stanford Alpaca with GPT-4 to calculate the win-rate. For models fine-tuned on Evol-instruction-66k, we use HumanEval (Chen et al., 2021) with pass@1 as evaluation metric. We use Exact Match (EM) for SQuAD and use accuracy (acc) for other tasks.

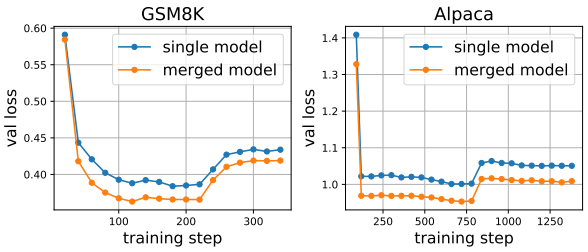

Figure 2: Comparison of validation loss between single and merged SFT models.

3.2 Experiments Across Different Models

Experiments were conducted using a variety of pre-trained models: BERT-base (0.11b)[4], BERT-large (0.34b), TinyLlama (1.1b), and Llama-2-7b (7b), employing the weighted-average method for

[3]For traditional tasks, experiments for decoder-based models use the version collected by Cheng et al. (2023); Wang et al. (2023). For MATH, an augmented version (Yu et al., 2023b) is used and data originally sourced from GSM8K is excluded. Evol-instruction-66k is obtained from https://huggingface.co/datasets/codefuse-ai/Evol-instruction-66k.

[4]‘0.11b’ refers to the model having 0.11 billion parameters.

Method	AG News acc	Hellaswag acc	MNLI acc	MRPC acc	SST-2 acc	Winogrande acc	Avg Δ
single SFT	94.42	77.20	87.90	85.78	95.53	75.45	-
weighted-avg	74.01	74.10	61.15	71.32	90.37	70.17	- 12.53
param-selection	77.03	**74.13**	64.77	67.16	92.66	70.40	- 11.67
. + resample	**81.28**	74.12	**64.45**	**72.55**	**95.30**	**70.56**	**- 9.67**

Table 3: Performance comparison of weighted-average and parameter-selection merging, based on Llama-2-7b, in the multi-task scenario for traditional tasks. ". + resample" means the addition of the resampling module to our parameter-selection method.

Method	GSM8K acc	MATH acc	AlpacaEval win-rate	HumanEval pass@1	Avg Δ
single SFT	63.76	14.26	89.29	23.78	-
weighted-avg	58.38	9.90	72.29	18.90	- 7.91
param-selection	57.01	10.1	72.08	14.64	- 9.32
. + resample	**61.71**	**11.7**	**78.70**	**26.22**	**- 3.19**

Table 4: Multi-task merging performance comparison for LLM tasks.

merging. Results, presented in Table 1, demonstrate that merged models outperform their single SFT counterparts. Additionally, we compared their validation loss, as illustrated in Figure 2: at all training checkpoints, the merged models exhibited lower loss compared to individual single SFT models. These experimental outcomes demonstrate the effectiveness of parameter merging in enhancing fine-tuning performance. Moreover, as detailed in Table 1, models with larger parameter sizes exhibit more pronounced average enhancements, suggesting the method's potential in large model contexts.

3.3 Weighted-Average vs Parameter-Selection

Experiments conducted based on Llama-2-7b are used to compare the weighted-average merging and parameter-selection merging. The results are presented in Table 2. As shown, compared to single SFT models, the merged models demonstrate performance improvements on three mainstream tasks (instruction-following, mathematical reasoning, and code-generating), except for a performance decrease observed with the weighted-average merging on the code-generating task. As indicated in Table 2, the parameter-selection outperforms the weighted-average method. Furthermore, by incorporating the resampling module, the performance parameter-selection can be further improved, yielding an average improvement of **2.02** percentage points across five datasets. These results demonstrate the efficacy of parameter-selection merging method proposed in this work.

3.4 Multi-Task Merging

We also executed experiments on integrating SFT models for multiple tasks to validate the effectiveness of the parameter-selection merging. For traditional tasks, experiments are conducted on Llama-2-7b with 6 text classification tasks collected by Cheng et al. (2023); Wang et al. (2023), and results are presented in Table 3. For LLM tasks, we use WizardLM-13B (instruction-following), WizardMath-13B (mathematical reasoning), and llama-2-13b-code-alpaca (code generating) (Chaudhary, 2023) as fine-tuned models,[5] wiht results detailed in Table 4. As shown in Table 3 and Table 4, parameter-selection method and the weighted-average method yield similar performance. However, when combined with the resampling module, the parameter selection method significantly outperforms the average-based method, achieving **2.86** and **4.72** more percentage points in performance retention on traditional and LLM tasks, respectively.

4 Conclusion

In this study, we enhanced LLM fine-tuning through merging sub-models fine-tuned on diverse data orders. Importantly, we introduced a novel merging technique: "parameter-selection merging" for the first time, which outperforms traditional weighted-average approach. The efficacy of parameter-selection method indicates that it is not necessary to incorporate information from all sub-models at each parameter dimension in parameter merging. This discovery broadens the research landscape for parameter merging, opening up new avenues for future investigations.

[5]Due to significant forgetting after merging LLM tasks, 13b models were chosen instead of 7b.

5 Limitations

There are several primary limitations in this study that remain unexplored:

- Although the parameter merging method enhances LLM fine-tuning without adding deployment and inference costs, it needs more computation to fine-tune multiple sub-models.
- The study mainly centers on merging sub-models trained on data sequences with different order. A broader range of configurations, such as merging sub-models trained with different learning rates, training steps, or batch sizes are unexplored. This limitation points to potential avenues for future research, where these variables can be systematically studied to understand their impact on parameter merging.
- The study introduces a novel technique: parameter-selection merging, that outperforms traditional weighted-average merging in the single-task scenario. However, many studies focus on merging multi-task models based on weighted-average formula. Whether substituting the weighted-average with parameter-selection can enhance these methods has not yet been explored.

References

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Denvy Deng, and Qi Zhang. 2023. Uprise: Universal prompt retrieval for improving zero-shot evaluation. *arXiv preprint arXiv:2303.08518*.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

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*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Learning to retrieve in-context examples for large language models. *arXiv preprint arXiv:2307.07164*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

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.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. 2023. Lm-cocktail: Resilient tuning of language models via model merging. *arXiv preprint arXiv:2311.13534*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Detailed Experimental Settings

A.1 Basic Settings

For all tasks, we uniformly train 20 sub-models for merging. For single SFT models, we report the average results across all sub-models. For parameter-selection merging models, we conduct 5 experiments with different random seeds and report the average outcomes. For decoder-based models, the temperature is set to 0.0 for greedy decoding. Training of LLMs was conducted using mixed precision bf16. All experiments are conducted on 8 NVIDIA Tesla A800 GPUs.

A.2 Hyperparameters

The search space for resampling times n includes {1 - 4, 9 }, corresponding to conducting random sampling 2 - 5, and 10 times in total, respectively. The hyperparameters used for fine-tuning are detailed in Tables 5 and 6. For multi-task merging, the resampling times n are set to 9 for traditional tasks and set to 2 for LLM tasks. For all experiments, the maximum number of epochs was set to 3, with model states saved at the end of each epoch. The checkpoint corresponding to the highest-performing epoch was reported. Parameter merging is performed only among models that have undergone the same number of training steps.

B Computational Complexity of Merging Process

The model merging process, whether through parameter selection or weighted averaging, can be managed on a CPU. Thanks to PyTorch's parallel processing capabilities, the merging process can be completed very fast: merging 10 Llama-2-7b models on a single CPU approximately takes about 2 minutes. The resampling process would require time proportional to the number of resampling iterations.

Model	BERT-base & BERT-large			TinyLlama & Llama-2-7b						
Dataset	SST-2	MNLI	SQuAD	SST-2	MNLI	SQuAD	AG News	Hellaswag	MRPC	Winogrande
max seq-length	128	128	512	800	800	800	800	800	800	800
learning rate	2e-5	2e-5	3e-5	2e-5	2e-5	2e-5	2e-5	2e-5	2e-5	2e-5
batch size	32	32	12	128	128	128	128	128	128	128

Table 5: Hyperparameters for training models on traditional tasks.

Dataset	AlpacaEval	GSM8K	GSM8K-RFT	MATH	HumanEval
max seq-length	1200	800	800	800	1200
learning rate	2e-5	2e-5	2e-5	2e-5	2e-5
batch size	128	64	64	64	128
max epoch	3	3	3	3	3
n	1	1	4	1	4

Table 6: Hyperparameters for training Llama-2-7b on LLM tasks.