Mitigating Training Imbalance in LLM Fine-Tuning via Selective Parameter Merging

Anonymous ACL submission

Abstract

Supervised fine-tuning (SFT) is crucial for adapting Large Language Models (LLMs) to specific tasks. In this work, we demonstrate that the order of training data can lead to significant training imbalances, potentially resulting in performance degradation. Consequently, we propose to mitigate this imbalance by merging SFT models fine-tuned with different data orders, thereby enhancing the overall effectiveness of SFT. Additionally, we introduce a novel technique, "parameter-selection merging," which outperforms traditional weighted-average methods on five datasets. Further, through analysis and ablation studies, we validate the effectiveness of our method and identify the sources of performance improvements.

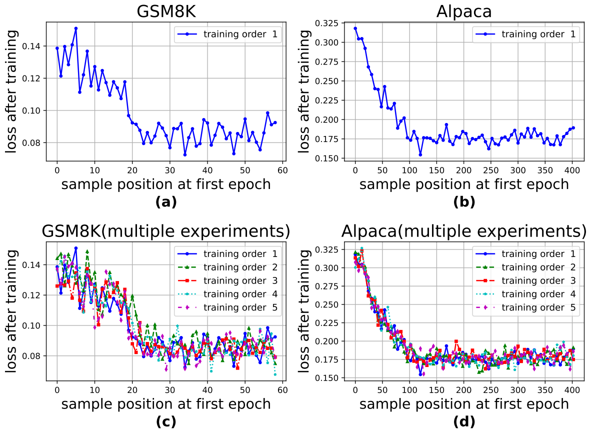

Figure 1: Impact of training sample position at first epoch on final model losses of these samples. Panels (a) and (b) present the results on the GSM8k and Alpaca tasks, respectively. Panels (c) and (d) show the corresponding results from multiple experiments with different training orders.

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).

The SFT process typically involves a few iterations of training on task-specific data. While existing research generally assumes that the order of training samples has a negligible impact on final model performance, or that sufficient iterations can mitigate any potential effects, our preliminary investigations suggest otherwise. We found that the position of SFT training samples significantly affects their final training outcomes. For instance, Figure 1 (a) and (b) illustrate the relationship between the position of training samples in the first epoch and their losses after three epochs of training. The figure clearly shows that despite multiple epochs of training, samples introduced earlier consistently exhibit higher final losses. Figure 1 (c) and (d) present the results of multiple experiments with different training orders, demonstrating a strong and consistent correlation between the position of training samples and their final losses.[1]

These findings suggest a notable imbalance in fine-tuning process: samples processed at different positions unevenly influence the learning process, thereby posing a potential risk of skewing the performance of the fine-tuned model. To mitigate this imbalance, we propose merging multiple SFT models obtained from diverse training data orders through parameter merging technique (Matena and Raffel, 2022). Moreover, we introduce "**parameter-selection merging**," a novel parameter merging method that outperforms the traditional weighted-average method. The core contributions

[1] The experiment was conducted using GSM8K (Cobbe et al., 2021) and Stanford Alpaca (Taori et al., 2023) datasets, with Llama-2-7b (Touvron et al., 2023) as the base model. Each epoch featured a different sample order.

of this paper are summarized as follows:

- We identify the training imbalance in SFT process, where the position of training samples significantly affects their final training losses.
- We propose to improve model fine-tuning by merging models trained with different data orders. Moreover, we introduce a novel parameter merging method, “parameter-selection merging."
- Through analysis and ablation studies, we further validate the effectiveness of our method and demonstrate the source of improvement.

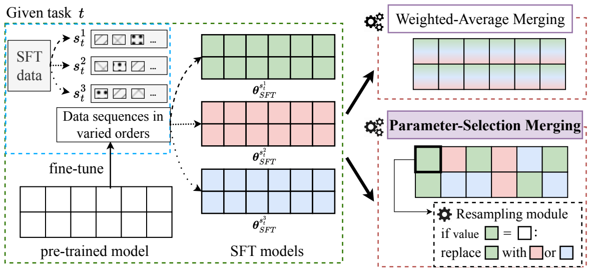

Figure 2: Illustration comparing weighted-average method and the proposed parameter-selection method. Weighted-average merging calculates the weighted sum of all sub-model parameters at each parameter dimension, whereas parameter-selection merging selects parameters from a single sub-model. In the resampling module, parameters that equal those of the base model are replaced with parameters from alternative models.

2 Method

2.1 Merge Fine-tuned LLMs with Different Data Order

In this work, we propose to mitigate training imbalance in LLM fine-tuning by merging models fine-tuned with various data orders. As depicted in Figure 2, 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 an improved SFT 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 2 shows the comparison of two merging techniques. Given a set of K sub-models $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K\}$, each model $\boldsymbol{\theta}_i$ is comprised of parameters $\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,d}$ across d parameter dimensions. Weighted-average merging calculates the weighted sum of all sub-model parameters at each parameter dimension, which can be represented by the following 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, as 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 that $\theta_{i,j}$ is selected. Given that each sub-model in our method is fine-tuned on the same training dataset and thus has nearly identical performance, we assign equal weights and selection probabilities among sub-models, set as: $w_i = \frac{1}{K}, p_i = \frac{1}{K}$.

2.3 Resample Strategy

Task Vectors. Let $\boldsymbol{\theta}_{\text{pre}}$ represent the pre-trained model’s weights and $\boldsymbol{\theta}_{\text{SFT}}$ denote the SFT model ’s weights. 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 intention to maximize the impact of task vectors, we introduce a resampling method within the parameter-selection merging framework to further improve task performance. $\tau_{i,j}$ represents the task vector of the i-th sub-model at parameter dimension j. As depicted in Figure 2, if $\tau_{i,j} = 0$, indicating that no parameter change occurred 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:

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

[2]This strategy enables parallel tensor operations by including all sub-models in resampling, not just the remaining ones.

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 1: 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.

Specifically, $\theta^0_{\text{merged},j}$ equals parameter-selection method without the resampling module.

3 Experiments

This section presents the experimental results. Detailed descriptions of the datasets and evaluation metrics employed are provided in the Appendix, under Section B.

3.1 Experimental Results

Main Experiments. We conducted experiments on three mainstream LLM tasks: instruction-following, mathematical reasoning, and code-generating. Llama-2-7b (Touvron et al., 2023) was used as the base model. As shown in Table 1, the merged models exhibit performance improvements compared to single SFT models. Furthermore, as indicated in Table 1, the proposed parameter-selection method outperforms the weighted-average approach, achieving consistent performance improvements. Moreover, incorporating a resampling module further enhances the performance of the parameter-selection method, yielding an average improvement of **2.02** percentage points across all datasets. These results affirm the effectiveness of our proposed method in improving LLM fine-tuning performance.

Experiments Across Different Model Sizes. We conducted experiments using different pre-trained models with various model sizes: BERT-base (0.11b)[3], BERT-large (0.34b) (Kenton and Toutanova, 2019), TinyLlama (1.1b) (Zhang et al., 2024), and Llama-2-7b (7b), employing parameter-selection as merging method.[4] As shown in Table 2, the merged models outperform their single SFT counterparts consistently. These experimental outcomes further demonstrate the effectiveness of merging SFT models with different training orders in improving fine-tuning performance. Furthermore, as detailed in Table 2, models with larger parameter sizes exhibit more pronounced average improvements, suggesting our method's potential applicability in LLM contexts.

Model	Method	SST-2 acc	MNLI acc	SQuAD EM	Avg Δ
BERT-base	SFT	91.93	83.99	81.07	+ 0.75
	merged	92.33 (+0.40)	**84.47**(+0.48)	**82.44**(+1.37)	
BERT-large	SFT	93.44	86.42	84.15	+ 0.94
	merged	**94.38**(+0.94)	86.71(+0.29)	**85.73**(+1.58)	
TinyLlama	SFT	94.81	85.46	80.53	+ 1.62
	merged	**95.91**(+1.10)	**86.93**(+1.47)	**82.82**(+2.29)	
Llama-2-7b	SFT	95.09	88.84	84.53	+ 2.11
	merged	**96.97**(+1.88)	**90.64**(+1.80)	**87.18**(+2.65)	

Table 2: Performance comparison between single SFT model and merged models across pre-trained models with various model sizes.

Experiments in Multi-Task Merging Contexts. We conducted experiments in multi-task merging contexts to validate the effectiveness of parameter-selection. Multi-task merging aims to combine single-task models into one multi-task model capable of handling several tasks simultaneously, with minimal performance loss in single-task capabilities.[5] As shown in Table 3, the parameter-selection method significantly outperforms the weighted-average method, achieving an increase of **4.72** percentage points in performance retention. This result demonstrates the efficacy of proposed parameter-selection method.

[3](0.11b) refers to the model having 0.11 billion parameters.

[4]Experiments were conducted on traditional tasks rather than on LLM tasks due to the limited capabilities of smaller-sized models.

[5]Due to significant performance degradation for LLM tasks, 13b models were chosen instead of 7b. We used WizardLM-13B (Xu et al., 2023), WizardMath-13B (Luo et al., 2023), and Llama-2-13b-code-alpaca (Chaudhary, 2023) as single SFT models for instruction-following, mathematical reasoning, and code-generating tasks, respectively.

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

Table 3: Performance comparison in multi-task merging contexts. The "single SFT" represents a single-task model, showing results for individual tasks, whereas the other entries are multi-task models, showing results for handling multiple tasks simultaneously.

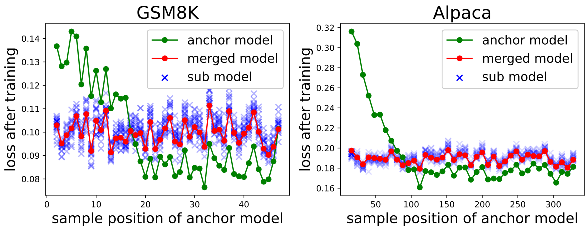

Figure 3: Comparison of training losses across different models, with the first epoch sample position of the anchor model as the x-axis. Green lines represent final training losses of the anchor model; blue 'x' markers indicate losses of SFT models trained with various data order; red dots show losses of the merged model.

3.2 Analysis and Ablation Studies

This section presents the analysis and ablation studies conducted on the GSM8K and Alpaca tasks.

Traning Set Loss Analysis. We investigate whether the merged models can alleviate the training imbalance problem previously identified. We selected one SFT model as the "*anchor model*". Based on positions during the first epoch training of the anchor model, we divided training samples into multiple segments. Figure 3 shows the final training loss of these sample segments. As shown in Figure 3, compared to the anchor model, the losses of the merged model are situated between those of sub-models, showing no clear correlation with the data position. This result indicates that merging models with various data orders can diminish the influence of the data order from a single model, such as the anchor model.

Validation Set Loss Analysis. We analyzed the validation set losses of the single SFT model and the merged model at various training steps. As

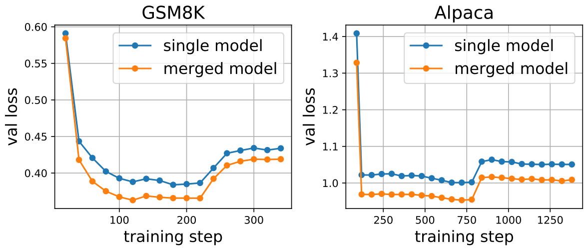

Figure 4: Comparison of validation loss between single and merged SFT models at various training steps.

Method	GSM8K acc	AlpacaEval win-rate
singel SFT	41.29	24.25
param-selection + resample	45.26	25.91
param-selection + resample (fix-batch)	45.51	25.83

Table 4: Performance comparison of standard merged models and models with fixed intra-batch combinations.

shown in Figure 4, at all training steps, the merged models exhibited lower validation losses compared to those of single SFT models. This result demonstrates that the merged model exhibits lower losses on unseen samples, which aligns with the performance enhancements previously observed.

Determining the Source of Improvement: Sample Position or Batch Diversity. Altering the order of training data not only changes the position of samples but also modifies the combinations of samples within each batch. This raises the question: Do performance improvements result from varied sample positions or from diversity in sample combinations? To address this, we conducted ablation experiments by merging models with fixed intra-batch sample combinations while varying batch positions. As shown in Table 4, models with fixed intra-batch combinations achieved similar performance to those with variable combinations, indicating that performance gains are primarily due to changes in sample positions rather than to diversity in intra-batch combinations.

4 Conclusion

This study highlighted how training data order affects LLM fine-tuning, leading to significant imbalances. Merging models with diverse data orders can mitigate these imbalances and improve model performance. Future research will focus on enhancing model robustness and extending parameter-selection merging technique to various scenarios.

Limitations

This study has several primary limitations that remain unexplored:

- While our method improves LLM fine-tuning without adding deployment and inference costs, it requires additional computation to fine-tune multiple sub-models.
- Although models with larger parameter sizes show more pronounced average improvements, as demonstrated in Table 2, suggesting the method's potential in LLM contexts, our experiments were primarily conducted with 7b models due to computational resource constraints. Future studies are needed to evaluate the scalability of our methods with larger models.
- The study introduces the novel parameter-selection merging technique, which outperforms the traditional weighted-average approach. However, many model merging studies in multi-task scenarios rely on a weighted-average formula. It remains to be explored whether replacing the weighted-average with parameter-selection can improve these existing methods in multi-task scenarios.

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*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

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*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

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*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

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 Related Work

A.1 Parameter Merging in Multi-Task Scenario

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 (multi-task scenario). 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.

A.2 Parameter Merging in Single-Task Scenario

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 improve single-task LLMs has not yet received much attention. While

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	512	512	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.

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 and achieved only modest improvements.

B Detailed Experimental Settings

B.1 Datasets

Datasets employed in our experiments are categorized into two groups: LLM tasks and traditional NLP tasks.

LLM Tasks:

- **Instruction-following:** Stanford Alpaca (Taori et al., 2023)
- **Mathematical Reasoning:** GSM8K (Cobbe et al., 2021), GSM8K-RFT (Yuan et al., 2023), MATH (Hendrycks et al., 2021)
- **Code-generating:** Evol-instruction-66k, obtained from Hugging Face Datasets

Traditional NLP Tasks:

- SST-2 (Xu et al., 2023)
- MNLI (Williams et al., 2017)
- SQuAD (Rajpurkar et al., 2016)
- AG News (Zhang et al., 2015)
- Hellaswag (Zellers et al., 2019)
- MRPC (Dolan and Brockett, 2005)
- Winogrande (Sakaguchi et al., 2020)

For traditional tasks, experiments involving decoder-based models utilized the version collected by Cheng et al. (2023); Wang et al. (2023). For the MATH dataset, an augmented version (Yu et al., 2023b) is employed, with data originally sourced from GSM8K excluded. The Evol-instruction-66k dataset is obtained from the Hugging Face library (https://huggingface.co/datasets/codefuse-ai/Evol-instruction-66k).

B.2 Evaluation Metrics

We employ AlpacaEval (Li et al., 2023) to evaluate models fine-tuned on Stanford Alpaca dataset, using win-rate as the evaluation metric and GPT-4 as the annotator. We employ HumanEval (Chen et al., 2021) to evaluate models fine-tuned on Evol-instruction-66k dataset, using pass@1 as the evaluation metric. For the SQuAD dataset, Exact Match (EM) is utilized as the evaluation metric. Accuracy (acc) is used as the evaluation metric for all other tasks.

B.3 Basic Settings

For single SFT models, we report the average results across all sub-models. For parameter-selection merging models, we conduct five 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 were conducted on 8

Method	AG News acc	Hellaswag acc	MNLI acc	MRPC acc	SST-2 acc	Winogrande acc	Avg Δ
			Single-Task Model				
single SFT	94.42	77.20	87.90	85.78	95.53	75.45	-
			Multi-Task Models				
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 7: Performance comparison in multi-task merging contexts for traditional tasks. ". + resample" refers to the addition of the resampling module to our parameter-selection method.

NVIDIA Tesla A800 GPUs.

B.4 Hyperparameters

For the parameter merging method, the number of sub-models K is a necessary hyperparameter. Based on the selection range of 1-50 suggested by Wortsman et al. (2022), we use $K = 20$, a relatively moderate value for all datasets (15 datasets in total). The search space for resampling times n includes $\{1, 2, 3, 4\}$. In our experiments, the maximum number of epochs was set to 3, with model states saved at the end of each epoch. The hyperparameters used for fine-tuning are detailed in Tables 5 and 6.

C Computational Complexity of Merging Process

The parameter selection and weighted-average merging processes can be efficiently managed on a CPU with rapid execution times. For instance, merging 10 Llama-2-7b models on a single CPU typically takes about 1 minute. The resampling process, meanwhile, requires time proportional to the number of resampling iterations n, with each iteration approximately taking about 0.1 minute.

D Experiments in Multi-Task Merging Contexts for Traditional Tasks

In multi-task merging contexts, we conduct experiments on six traditional tasks using Llama-2-7b as the base model. The results are presented in Table 7. Consistent with the results for LLM tasks, the parameter-selection method outperforms the average-based method as well, achieving **2.86** more percentage points in performance retention.