Boosting Large Language Models with Mask Fine-Tuning

Anonymous ACL submission

Abstract

The large language model (LLM) is typically integrated into the mainstream optimization protocol. No work has questioned whether maintaining the model integrity is *indispensable* for promising performance. In this work, we introduce Mask Fine-Tuning (MFT), a novel LLM fine-tuning paradigm demonstrating that carefully breaking the model's structural integrity can surprisingly improve performance without updating model weights. MFT learns and applies binary masks to well-optimized models, using the standard LLM fine-tuning objective as supervision. Based on fully fine-tuned models, MFT uses the same fine-tuning datasets to achieve consistent performance gains across domains and backbones (e.g., an average gain of **2.70 / 4.15** in IFEval with LLaMA2-7B / 3.1-8B). Detailed ablation studies and analyses examine the proposed MFT from different perspectives, such as sparse ratio and loss surface. Additionally, by deploying it on well-trained models, MFT is compatible with collaborating with other LLM optimization procedures to enhance the general model. Furthermore, this study extends the functionality of the masking operation beyond its conventional network-pruning context for model compression to a broader model capability scope.

1 Introduction

Pre-training large language models (LLMs) typically requires massive corpora to enable models with foundational knowledge and linguistic competencies for effective language generation (Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023; Grattafiori et al., 2024). Afterward, the pre-trained LLMs are fine-tuned for downstream tasks using high-quality domain data, including specialized knowledge such as math and coding (Cobbe et al., 2021; Yu et al., 2024; Chen et al., 2021; Austin et al., 2021) and particular patterns such as human instruction (Zhou et al., 2023; Dubois et al., 2024). For such adaptations, fine-tuning all model parameters is the most straightforward and common approach, commonly referred to as full fine-tuning (FFT), and generally yields the most competitive performance. Furthermore, adapter-based alternatives offer parameter-efficient fine-tuning (PEFT) strategies such as vanilla bottleneck adapters (Houlsby et al., 2019) and LoRA (Hu et al., 2022). These methods fix the pre-trained backbone and additionally introduce a limited number of learnable parameters. Even if such methods may sacrifice fine-tuning performance relative to FFT, they are well-suited to scenarios with limited computational budgets, especially when fine-tuning data are insufficient. Overall, the pipeline of language model pre-training followed by fine-tuning has demonstrated promising results across a range of language tasks, including domains requiring extensive specialized knowledge (e.g., medicine (Wang et al., 2023) and law (Nguyen, 2023)) and complex reasoning (e.g., math (Frieder et al., 2024) and coding (Chen et al., 2021)), exhibiting remarkably intelligent capabilities.

However, the mainstream LLM optimization process (pre-training followed by fine-tuning), which is promisingly benchmarking modern LLM performance, consistently treats the language model as a whole. For pretraining, the model's structure is kept mostly dense. For fine-tuning, FFT optimizes all model parameters simultaneously, and PEFT freezes it to tune additional adapter parameters, where both accept the necessity of LLM structural integrity by default. Such practice naturally inspires us to ask: *Is such structural integrity indispensable for good model performance?*, or even further, is there potential to further improve the model by removing certain model components that break the structural integrity?

We propose *mask fine-tuning (MFT)* to answer this question. MFT freezes a given LLM model and learns a binary mask over it to assess whether break-

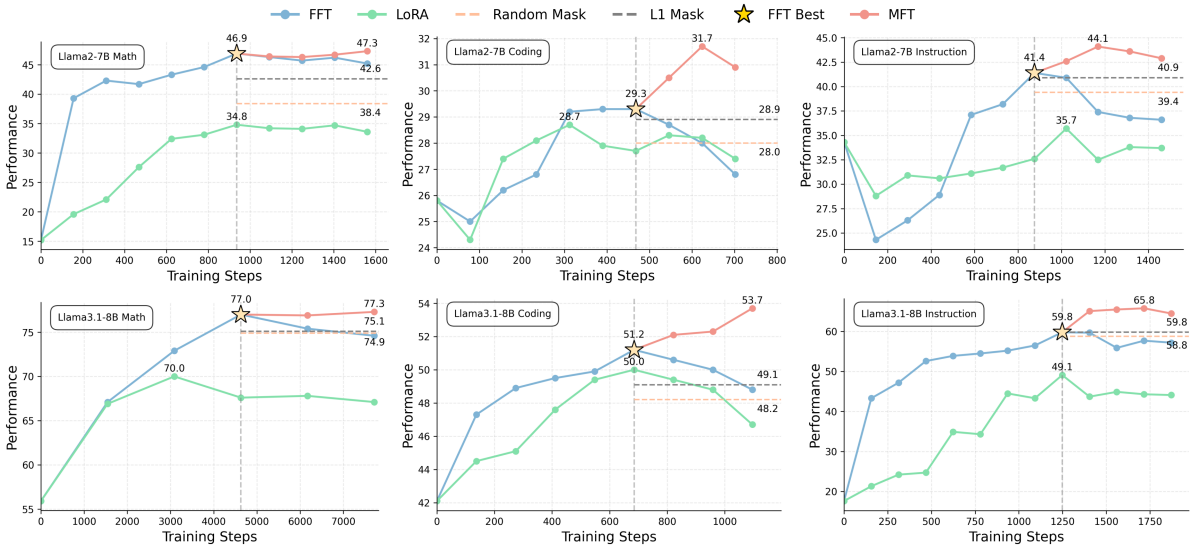

Figure 1: The visualization of the performance trend across different fine-tuning strategies, including FFT (blue line), LoRA (green line) (Hu et al., 2022), and our MFT (red line). We also add random (orange dashed) and L1 (gray dashed) masks for comparison. We use three settings across LLaMA2 and LLaMA3.1 backbones on GSM8K, HumanEval, and IF-Eval for math, coding, and instruction domains, respectively. The x-axis is training steps starting from the pre-trained backbone. The y-axis is evaluation performance. MFT (red line) starts from the best FFT model (yellow star) and improves upon the upper bound, whereas continued FFT leads to overfitting. It also outperforms LoRA fine-tuning and two vanilla mask baselines.

ing structural integrity works. To substantiate its effectiveness, we use a well-trained LLM (model after sufficient FFT in this study, serving as a strong representative baseline) as the starting point to assess whether MFT yields further improvement. MFT uses the standard LLM fine-tuning objective and datasets to learn the mask, which are the same as those for the FFT. The only difference is that the learnable part is changed from model weights to a binary mask of the weights, while the weights themselves are fixed. The learned mask indicates positions of certain parameters to be removed. Surprisingly, we find that masking these weights yields further performance gains compared with the given well-trained model. As shown in Fig. 1, the regular FFT (blue line) improves the pre-trained model across different settings, but once the best performance is reached (yellow star), excessive FFT leads to overfitting. However, MFT (red line) starts from the best FFT model and further improves it. These teaser cases show a consistent observation across different backbones (LLaMA2-7B (Touvron et al., 2023) and LLaMA3.1-8B (Grattafiori et al., 2024)) and domains (GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and IF-Eval (Zhou et al., 2023) for math, coding, and instruction, respectively). Comparisons of more configurations are elaborated in Sec. 3. Such a phenomenon substantially answers the question we are interested in: *the LLM structural integrity is not indispensable for good model performance, and breaking such integrity can lead to further improvements.*

Here, we primarily explore MFT as a post-fine-tuning strategy for further improvement, starting from a yet outperforming FFT checkpoint. It naturally upgrades the current fine-tuning routine into a new protocol, as shown in Fig. 2, providing a new perspective for investigating LLM fine-tuning. Furthermore, MFT is compatible with existing fine-tuning methods and can be integrated flexibly into any of them, as it shares the same optimization objective and requires no additional data annotation.

Besides, we emphasize the difference between typical pruning methods and our work. The former compresses the model and aims to preserve the trained model's capabilities, whereas the latter seeks further improvement beyond a well-trained model without a specific purpose. Even if they share the same concept of *mask* (or model sparsity), their fundamental goals are different (see more discussions in Sec. 4). Following this line, our study extends the functionality of the masking operation beyond typical network compression to a more general model-capability scenario. In other words, analogically speaking, *typical masking employs subtraction to reduce (pruning to compress), whereas our approach leverages subtraction to achieve augmentation (removing weights to*

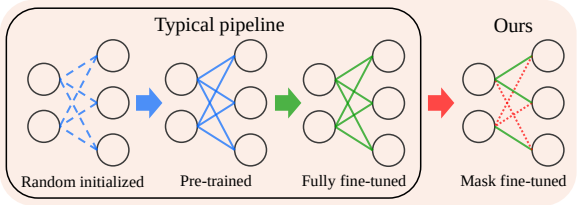

Figure 2: Typical LLM training comprises pre-training and fine-tuning to build capacity and domain knowledge, with the structure always being entire. We are curious if such integrity is necessary and propose MFT to generally outperform models with sufficient FFT, naturally upgrading the classic pipeline following the typical protocol to further refine well-optimized LLMs.

improve). Our contributions are as follows:

- We validate that a well-trained LLM can be further improved by carefully removing certain weights using the same objective and dataset as regular fine-tuning, with limited computational overhead. Such a strategy is compatible with other training pipelines and enhances the current LLM optimization.
- We propose *Mask Fine-Tuning (MFT)* to complete our exploration in a post fine-tuning scenario. MFT starts from a competitive, fully fine-tuned model with fixed weights and learns a binary mask that is applied to the model to improve it. Following this, MFT treats model sparsity in a new light, not only for efficiency, but also for performance improvement.
- Extensive experiments across different backbones (LLaMA2 and LLaMA3.1), domains (math, coding, and instruction), and FFT settings (domain-specific and mixed-up) show the effectiveness of MFT with consistent performance gain. Detailed ablations and analyses are provided to provide better intuition and inspire future work.

2 Mask Fine-Tuning

We focus on GPT-like language models (Brown et al., 2020; Radford et al., 2019) in an auto-regressive manner and start from a fully fine-tuned model of a pre-trained language backbone to perform mask fine-tuning (MFT). Therefore, we briefly introduce the language model notation for full fine-tuning (FFT), and then present our MFT.

Full Fine-Tuning. We refer a pre-trained auto-regressive language backbone as $\mathcal{N}_p$ with parameters Θ_p and optimize the following objective to represent a FFT:

$$L(U_f) = \sum_i logP(u_f^i | u_f^{i-k}, ..., u_f^{i-1}; \Theta_p), \quad (1)$$

where L represents the auto-regressive loss to supervise the next token prediction. Θ_p is fully optimized, and $U_f = \{u_f^1, ..., u_f^n\}$ is a token sequence of a language corpus, serving as an FFT dataset. We refer to the model obtained after sufficient FFT as $\mathcal{N}_f$ and denote its optimized parameters by Θ_f.

Mask Fine-Tuning. We deploy our MFT on models after the FFT. Formally, we revise Eq. (1) by adding a binary mask onto model parameters and obtain the rewritten loss for MFT as below:

$$L(U_m) = \sum_i logP(u_m^i | u_m^{i-k}, ..., u_m^{i-1}; \Theta_f \odot M), \quad (2)$$

where it shares the same objective function L as in FFT, using the dataset U_m for supervision. Herein, the dataset U_m for MFT is the same as U_f for FFT in our experiments. The main difference is that we add a binary mask M on the given model parameters Θ_f, where M and Θ_f share the same size and corresponding weights on Θ_f are masked out by element-wise multiplication $\odot$. During MFT, Θ_f is fixed, and M is updated. This way, MFT learns to locate specific weights while keeping the fully fine-tuned parameters Θ_f unchanged. These weights can be removed from the well-trained model by applying the learned binary mask. M is optimized using a learnable process based on the straight-through gradient estimator (Bengio et al., 2013), which we call the learnable mask.

Learnable Mask. Given a model parameter Θ_f after FFT, we represent it in a layer-wise fashion as $\Theta = \{\theta_l\}, l \in \{1, 2, ..., L\}$, where we omit the subscript f for notation simplicity. L represents the model layer number. Then, we can formalize each layer as

$$I_{l+1} = \sigma(F[I_l; \theta_l]), \quad (3)$$

where I_l is the layer l input and I_{l+1} is its output with activation σ. F generally serves as a layer operation (e.g., a convolutional or linear layer) with a parameter $\theta_l = \{\theta_l^d\}, d \in \{1, 2, ..., D_l\}$. D_l represents the layer l parameter dimension. In this work, F denotes a linear layer, as we primarily consider Transformer-based language models and apply the learnable mask to the model's linear mappings. To perform mask learning, we fix all parameters Θ, denoted as $\overline{\Theta}$. Then, each weight $\overline{\theta_l^d}$ is assigned with a learnable score c_l^d, representing the importance of the corresponding weight. Based on this, we rewrite Eq. (3) as

$$I_{l+1} = \sigma(F[I_l; \overline{\theta_l} \odot v(c_l)]), \quad (4)$$

where $c_l = \{c_l^1, c_l^2, ..., c_l^{D_l}\}$ is the score of parameter $\overline{\theta_l}$. v is an indicator function to select a mask based on scores. In this work, we use a ratio-based version in which v outputs 1 if c_l^d is among the top $K\%$ highest-scoring values, and 0 otherwise, where K is a pre-defined mask ratio. By updating c with fixed $\overline{\Theta}$, part of the parameters in $\overline{\Theta}$ are kept while others are masked out. As the indicator function v is non-differentiable, we use the straight-through gradient estimator (Bengio et al., 2013) to estimate the gradient of the loss with respect to c_l^d. Concretely, v is regarded as an identity function during the gradient backwards step, then the approximation can be described as

$$\tilde{g}(c_l^d) = \frac{\partial L}{\partial \tilde{I}_{l+1}} \frac{\partial \tilde{I}_{l+1}}{\partial c_l^d} \approx \frac{\partial L}{\partial I_{l+1}} \frac{\partial I_{l+1}}{\partial c_l^d}, \quad (5)$$

where $\tilde{I}_{l+1} = \sigma(F[I_l; \overline{\theta}_l \odot c_l])$ after applying estimation and $\tilde{g}(c_l^d)$ is the estimated gradient with respect to score c_l^d. In this way, MFT learns to obtain a mask for the model $\mathcal{N}_f$, and we denote the resulting model as $\mathcal{N}_m$ with parameters Θ_f and mask M.

MFT learns to identify specific weights guided by the regular training loss of language models and then removes them by applying the learned mask. It answers our question (Sec. 1) and provides a new LLM fine-tuning protocol. We illustrate the MFT procedure in Fig. 2. Please note, the FFT method used in this study is supervised fine-tuning (SFT) without considering other policy-based tuning like DPO (Direct Preference Optimization) (Rafailov et al., 2024) and PPO (Proximal Policy Optimization) (Schulman et al., 2017). Our MFT can be easily generalized to them, and we leave such explorations to our future work.

3 Experiments

We evaluate the proposed mask fine-tuning (MFT) on large language models (LLMs) across several backbones, domains, and full fine-tuning (FFT) settings. In addition, we provide ablation studies and visualizations to provide better intuition.

3.1 Experimental Setups

Pretrained Backbones. We use Transformer (Vaswani, 2017) based pretrained LLMs as backbone models, including LLaMA2-7B (Touvron et al., 2023) and LLaMA3.1-8B (Grattafiori et al., 2024).

Domains & Datasets. We involve three domains with their representative tasks. Specifically, we include GSM8K (Cobbe et al., 2021) and MetaMath (Yu et al., 2024) for math, HumanEval and HumanEval+ (Chen et al., 2021) for coding, and IF-Eval (Zhou et al., 2023) and Alpaca-Eval (Dubois et al., 2024) for instruction following domains, respectively. All three domains are used for the typical FFT, our MFT, and evaluation.

Configurations. Our experiments use pretrained backbones, with FFT, MFT, and downstream evaluation in that order. For FFT, we follow two strategies: 1) domain-specific, 2) mix-up, where the former one conducts FFT using datasets from individual domains, such as math, coding, or instruction following, and the latter one uses a mix-up dataset from all these domains. For MFT, we follow a domain-specific approach, using datasets from a single domain. Further details of the dataset are provided in the Appendix. We mainly investigate local masking in this paper, which is elaborated by a series of proof-of-concept studies in Sec. 3.2 and Fig. 3. We also provide an initial exploration of global masking strategy in Sec. 3.3 and Tab. 6. For evaluation, we comprehensively test performance changes within (Tab. 1 and Tab. 2).

Baselines. We use several baselines for comparisons, including (1) a very basic pre-trained backbone; (2) FFT as a strong baseline, which is mostly seen as the most competitive approach for downstream fine-tuning. We perform a sufficient number of FFTs and report the best performance; (3) LoRA (Hu et al., 2022) fine-tuning as a strong alternative of FFT; (4) continued FFT, which always causes a performance drop due to overfitting; (5) continued LoRA plays a similar role for the LoRA baseline. We train all baselines for 4 epochs to ensure sufficient training and report the final performance for Continued FFT and Continued LoRA. (6) random mask, and (7) L1 mask as two vanilla masking strategies. Among them, (2) FFT and (4) continued FFT are critical to our experiments, since our goal is to use MFT to further improve the best FFT, and continued FFT serves as a sanity check.

3.2 Main Results

As mentioned in Sec. 3.1, MFT primarily uses a local masking strategy. Therefore, we begin with a proof-of-concept study to assess whether MFT has potential for further improvements (Fig. 3). Such ablation also helps us identify the most effective components of the well-trained model for MFT.

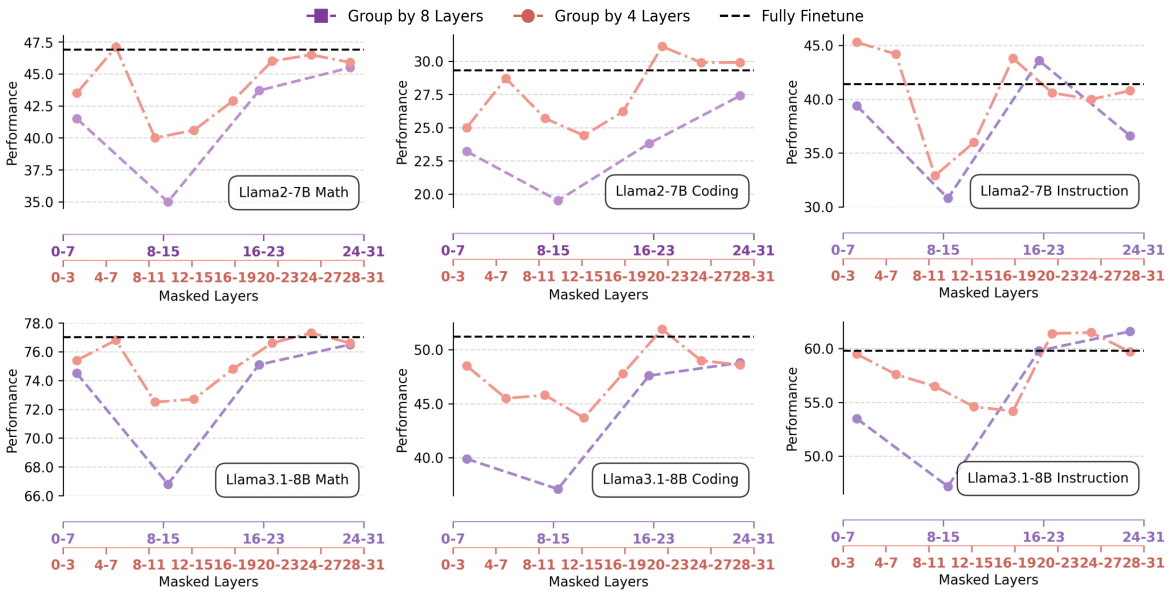

Figure 3: Visualization of ablation study of local MFT strategy. It uses LLaMA2-7B and LLaMA3.1-8B backbones, covering math, coding, and instruction domains. In each figure, we conduct MFT ablations with a 10% masking ratio on a domain-specific FFT model (black dashed line). We swap the ablation between two local granularities, 8-layer (purple) and 4-layer (orange), moving from shallow to deep layers using a 10% fine-tuning set. We find 1) MFT can outperform the FFT strong baseline and 2) MFT performs better in shallow (0-7) and relatively deep layers (20-27). This ablation is intended for quick intuition, with limited performance gains from MFT on a 10% subset. Based on this trend, we deploy MFT with complete fine-tuning sets, achieving more improvements.

Accordingly, we extensively deploy MFT in this local fashion across different configurations and compare with strong fine-tuning baselines.

Proof-of-Concept Study of Local MFT. This series of ablations aims to assess whether the MFT can improve the model's capability after FFT. If so, we expect a comprehensive ablation to also detect the sensitivity of different model components to MFT. Specifically, we start from the LLaMA2-7B and LLaMA3.1-8B backbones after domain-specific FFT and perform MFT on a small subset of each domain. Specifically, for each domain, we use 10% of its corresponding fine-tuning set. To swap the model, we group 4 or 8 layers in a row to ablate the model from shallow to deep layers. For each group, we use MFT to locally learn a mask and keep the other layers fixed to assess model performance. For example, for layers 4-7, this is formally achieved by setting $\Theta = \{\theta_l\}$, where $l \in \{4, 5, 6, 7\}$. Within each group, we set the masking ratio to 10% to learn the mask, meaning we aim to learn and remove 10% of the parameters while keeping the remaining 90%.

Fig. 3 shows the detailed ablation results. We find that 1) compared with FFT (black dashed line), the MFT has the potential to further improve for each setting, even if they may have different proper layer-wise groups for MFT; 2) A smaller layer-wise group (4-layer) is better for MFT compared with a larger one (8-layer), however, they generally share a similar trend, where sçhallow (0-7) and mid-to-rear (20-27) partitions often respond positively to MFT than other parts. Such observation confirms the MFT's potential to further improve models after FFT and indicates promising partitions for mask learning. Concretely, for LLaMA2-7B, we use 4-7, 20-23, and 0-3 for math, coding, and instruction domains, respectively, for MFT. For LLaMA3.1-8B, we use 24-27, 20-23, and 24-27 for the three domains, respectively. For all full-dataset MFT results reported in the rest of this paper, we follow the proof-of-concept study and use a constant 10% sparse ratio.

Notably, 1) Since this ablation is only for quick intuition and sanity check of MFT effectiveness, we only use a 10% dataset subset here. According to such insights, MFT is trained using complete fine-tuning datasets for more performance gain; 2) Since larger layer groups (e.g., 16) achieve a significant performance drop, we omit their results for a better illustration and leave them in the Appendix.

Comparisons. Tab. 1 and Tab. 2 present our main comparisons using the LLaMA2-7B and LLaMA3.1-8B backbones. We start from a pre-trained model, followed by two FFT settings using the fine-tuning dataset of specific and mixed

Table 1: Performance comparison on LLaMA2-7B of domain-specific and multi-domain mix-up FFT settings. Each part has three blocks containing 1) common LoRA and its continued variant (green), 2) FFT serving as an upper bound and its continued variant (blue), and 3) our MFT (red) with the other two vanilla masking baselines. The pre-trained model's performance is shown at the top and serves as a lower bound in our evaluation.

	Method	Math		Coding		Instruction Following	
		GSM8K	Math	HumanEval	HumanEval+	IF-Eval	Alpaca-Eval
	Pre-Trained Model	15.2	2.5	25.8	22.4	34.3	0.5
Specific Domain	Best LoRA	$34.8_{\pm0.34}$	$4.7_{\pm0.26}$	$28.7_{\pm0.56}$	$23.8_{\pm0.48}$	$35.7_{\pm0.68}$	$1.2_{\pm0.16}$
	Continued LoRA	$33.6_{\pm0.42}$	$4.5_{\pm0.23}$	$28.2_{\pm0.49}$	$23.2_{\pm0.43}$	$32.5_{\pm0.84}$	$1.1_{\pm0.12}$
	Best FFT	$46.8_{\pm0.22}$	$6.6_{\pm0.21}$	$29.4_{\pm0.36}$	$25.1_{\pm0.34}$	$41.2_{\pm0.62}$	$1.9_{\pm0.18}$
	Continued FFT	$45.0_{\pm0.28}$ ↓ 1.8	$5.5_{\pm0.23}$ ↓ 1.1	$27.9_{\pm0.42}$ ↓ 1.5	$23.6_{\pm0.38}$ ↓ 1.5	$37.8_{\pm0.68}$ ↓ 3.4	$2.0_{\pm0.21}$ ↑ 0.1
	Random Mask w/ Best FFT	$38.4_{\pm0.38}$	$4.9_{\pm0.25}$	$28.0_{\pm0.51}$	$24.0_{\pm0.47}$	$39.4_{\pm0.99}$	$1.4_{\pm0.21}$
	L1 Mask w/ Best FFT	42.6	5.7	28.9	24.5	40.9	1.5
	MFT w/ Best FFT **(Ours)**	**47.3**$_{\pm0.19}$ ↑ 0.5	**7.4**$_{\pm0.24}$ ↑ 0.8	**31.8**$_{\pm0.31}$ ↑ 2.4	**27.9**$_{\pm0.25}$ ↑ 2.8	**44.1**$_{\pm0.72}$ ↑ 2.9	**3.0**$_{\pm0.22}$ ↑ 1.1
Mixed Domain	Best LoRA	$40.8_{\pm0.30}$	$6.1_{\pm0.28}$	$22.6_{\pm0.55}$	$18.3_{\pm0.44}$	$37.3_{\pm0.81}$	$0.7_{\pm0.11}$
	Continued LoRA	$31.9_{\pm0.24}$	$4.0_{\pm0.21}$	$20.1_{\pm0.42}$	$17.1_{\pm0.36}$	$31.5_{\pm0.92}$	$0.8_{\pm0.09}$
	Best FFT	$45.5_{\pm0.18}$	$8.1_{\pm0.26}$	$29.7_{\pm0.43}$	$26.7_{\pm0.38}$	$43.6_{\pm0.68}$	$1.0_{\pm0.15}$
	Continued FFT	$44.1_{\pm0.22}$ ↓ 1.4	$7.5_{\pm0.27}$ ↓ 0.6	$21.1_{\pm0.58}$ ↓ 8.6	$18.1_{\pm0.43}$ ↓ 8.6	$38.6_{\pm0.74}$ ↓ 5.0	$1.2_{\pm0.16}$ ↑ 0.2
	Random Mask w/ Best FFT	$40.0_{\pm0.33}$	$6.5_{\pm0.32}$	$23.7_{\pm0.57}$	$19.1_{\pm0.49}$	$30.0_{\pm0.95}$	$0.9_{\pm0.14}$
	L1 Mask w/ Best FFT	43.3	7.7	25.7	22.3	32.8	1.2
	MFT w/ Best FFT **(Ours)**	**45.9**$_{\pm0.17}$ ↑ 0.4	**8.4**$_{\pm0.28}$ ↑ 0.3	**31.5**$_{\pm0.46}$ ↑ 1.8	**27.3**$_{\pm0.41}$ ↑ 0.6	**46.1**$_{\pm0.71}$ ↑ 2.5	**1.9**$_{\pm0.18}$ ↑ 0.9

Table 2: Performance comparison on LLaMA3.1-8B of domain-specific and multi-domain mix-up FFT settings. Each part has three blocks containing 1) common LoRA and its continued variant (green), 2) FFT serving as an upper bound and its continued variant (blue), and 3) our MFT (red) with the other two vanilla masking baselines. The pre-trained model's performance is shown at the top and serves as a lower bound in our evaluation.

	Method	Math		Coding		Instruction Following	
		GSM8K	Math	HumanEval	HumanEval+	IF-Eval	Alpaca-Eval
	Pre-Trained Model	55.9	14.6	42.1	37.8	17.6	9.0
Specific Domain	Best LoRA	$70.0_{\pm0.28}$	$22.7_{\pm0.22}$	$50.0_{\pm0.32}$	$45.1_{\pm0.43}$	$49.1_{\pm0.92}$	$10.1_{\pm0.51}$
	Continued LoRA	$67.1_{\pm0.35}$	$19.7_{\pm0.19}$	$46.7_{\pm0.46}$	$44.5_{\pm0.38}$	$44.3_{\pm0.88}$	$8.9_{\pm0.29}$
	Best FFT	$76.8_{\pm0.18}$	$24.2_{\pm0.18}$	$51.0_{\pm0.39}$	$44.9_{\pm0.31}$	$59.6_{\pm0.69}$	$11.9_{\pm0.36}$
	Continued FFT	$74.4_{\pm0.17}$ ↓ 2.4	$23.9_{\pm0.17}$ ↓ 0.3	$48.6_{\pm0.36}$ ↓ 2.4	$41.9_{\pm0.30}$ ↓ 3.0	$57.5_{\pm0.67}$ ↓ 2.1	$11.3_{\pm0.34}$ ↓ 0.6
	Random Mask w/ Best FFT	$74.9_{\pm0.37}$	$22.3_{\pm0.23}$	$48.2_{\pm0.48}$	$42.7_{\pm0.41}$	$58.8_{\pm0.96}$	$11.3_{\pm0.48}$
	L1 Mask w/ Best FFT	75.1	22.7	49.1	44.1	59.8	11.5
	MFT w/ Best FFT **(Ours)**	**77.3**$_{\pm0.14}$ ↑ 0.5	**24.8**$_{\pm0.21}$ ↑ 0.6	**53.5**$_{\pm0.41}$ ↑ 2.5	**46.8**$_{\pm0.40}$ ↑ 1.9	**65.6**$_{\pm0.70}$ ↑ 6.0	**13.7**$_{\pm0.32}$ ↑ 1.8
Multi Domain	Best LoRA	$72.8_{\pm0.42}$	$24.5_{\pm0.29}$	$63.1_{\pm0.82}$	$55.5_{\pm0.65}$	$38.2_{\pm1.18}$	$11.0_{\pm0.64}$
	Continued LoRA	$70.4_{\pm0.36}$	$24.1_{\pm0.27}$	$59.1_{\pm0.75}$	$54.3_{\pm0.59}$	$32.1_{\pm1.05}$	$5.3_{\pm0.32}$
	Best FFT	$72.7_{\pm0.16}$	$24.4_{\pm0.21}$	$64.4_{\pm0.59}$	$60.2_{\pm0.48}$	$60.0_{\pm0.73}$	$11.8_{\pm0.48}$
	Continued FFT	$71.7_{\pm0.14}$ ↓ 1.0	$24.2_{\pm0.20}$ ↓ 0.2	$62.6_{\pm0.53}$ ↓ 1.8	$58.3_{\pm0.44}$ ↓ 1.9	$59.7_{\pm0.80}$ ↓ 0.3	$8.8_{\pm0.38}$ ↓ 3.0
	Random Mask w/ Best FFT	$72.5_{\pm0.39}$	$23.2_{\pm0.28}$	$58.5_{\pm0.73}$	$55.5_{\pm0.62}$	$57.7_{\pm1.22}$	$10.0_{\pm0.51}$
	L1 Mask w/ Best FFT	72.5	24.1	63.4	59.1	58.9	10.7
	MFT w/ Best FFT **(Ours)**	**73.6**$_{\pm0.20}$ ↑ 0.9	**25.2**$_{\pm0.23}$ ↑ 0.8	**66.3**$_{\pm0.51}$ ↑ 1.9	**61.4**$_{\pm0.50}$ ↑ 1.2	**62.3**$_{\pm0.69}$ ↑ 2.3	**12.2**$_{\pm0.34}$ ↑ 0.4

domains for each backbone. For each setting, we perform sufficient FFTs and report the best and continued versions (blue block), and we do the same for LoRA as well (green block). We also include random and L1 masks as two vanilla masking baselines (white block). MFT (red block) starts from the best-performing FFT and further improves performance. All three masking methods are based on the best FFT as a strong starting point. Among them, two critical baselines are the best FFT and continued FFT, where the former serves as a competitive baseline and the latter as a sanity check for potential performance drop due to overfitting, thereby demonstrating MFT's effectiveness. We train our MFT only using domain-specific datasets for both FFT scenarios (domain-specific and mixed-up). Additionally, we compare the training cost in Fig. 4 with respect to GPU memory usage, the number of tokens used, and overall training time on A100 GPUs. We use the coding domain as an example. Additional comparisons are provided in the Appendix. Compared with the most important baseline, Best FFT, MFT involves very limited overhead for both token usage and training time. Compared with the Continued FFT, which also incurs at least the same training overhead, and possibly more, the Best FFT and MFT incur a lower training cost while achieving performance improve-

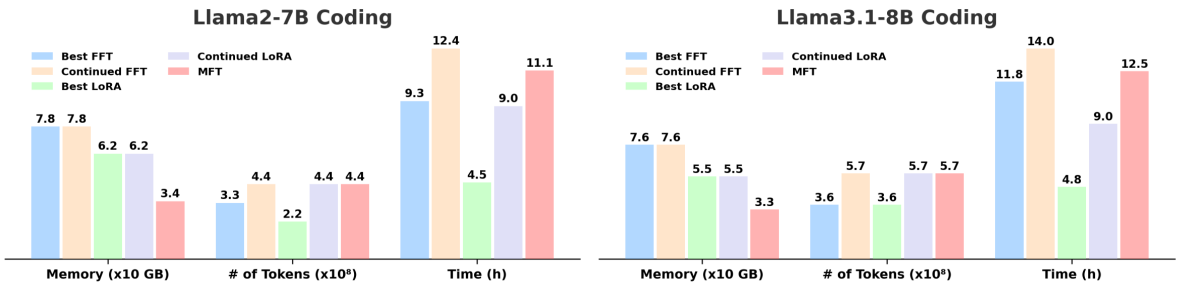

Figure 4: Training cost comparisons of GPU memory, used token number, and training time aspects. We use the coding domain as an example to compare MFT with other FFT and LoRA baselines.

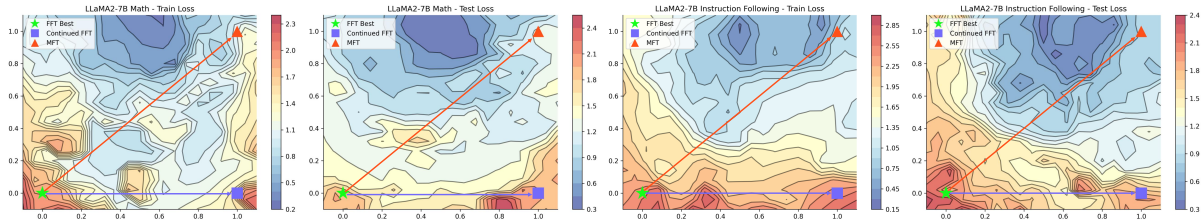

Figure 5: The loss landscape visualization on math and instruction following domains using LLaMA2-7B. Such visualization indicates that the proposed MFT further refines the Best FFT model, improving optimization and generalization.

ments. Since MFT fixes all model parameters and performs only local mask learning, it always has an advantage in memory usage.

Accordingly, we conclude: 1) Since the Continued FFT causes a performance drop and the Best FFT outperforms other comparison methods, it is reasonable to set it as our strong baseline. 2) MFT achieves promising and consistent improvements across different backbones, domains, and FFT configurations. It accordingly upgrades the current routine as a new LLM fine-tuning protocol by learning and applying a mask on a well-trained model. Moreover, the other two masking baselines generally degrade performance, especially compared with the consistent improvements achieved by MFT, indicating that MFT is not a trivial process and warrants further exploration.

3.3 Analyses

Theoretical Analysis. We follow PAC-Bayes theory for a generalization upper bound (McAllester, 1998), which has also been widely explored for neural networks (Arora et al., 2018), to provide theoretical intuition from an information-theoretic perspective. We focus on the performance gain from the model after the Best FFT to the model after MFT. Since our MFT uses the same training objective and dataset as FFT, the analysis is in a fair comparison, allowing a purely model generalization upper bound.

Given n as the size of the training set, δ as the degree of confidence, for any hypothesis h, which represents the model here, we have the PAC-Bayes upper bound given by

$$L(h) \leq L_S(h) + \Phi(C(h)), \quad \Phi(u) = \sqrt{\frac{u + \ln \frac{1}{\delta}}{2(n-1)}}. \tag{6}$$

$L(h)$ and $L_S(h)$ represent training and test loss, respectively. $\Phi(C(h))$ is an additional complexity term to describe the model code length, where $C(\cdot)$ represents the encoding process. Then, we have

$$\begin{aligned} &L(h_{\text{FFT}}) \leq L_S(h_{\text{FFT}}) + \Phi(C_{\text{FFT}}) \\ &\Rightarrow U(h_{\text{FFT}}) = L_S(h_{\text{FFT}}) + \Phi(C_{\text{FFT}}), \end{aligned} \tag{7}$$

$$\begin{aligned} &L(h_{\text{MFT}}) \leq L_S(h_{\text{MFT}}) + \Phi(C_{\text{MFT}}) \\ &\Rightarrow U(h_{\text{MFT}}) = L_S(h_{\text{MFT}}) + \Phi(C_{\text{MFT}}), \end{aligned} \tag{8}$$

where $U(\cdot)$ means the loss upper bound of h. If we make a difference on two sides, we have

$$\begin{aligned} &U(h_{\text{MFT}}) - U(h_{\text{FFT}}) \\ &= \left[L_S(h_{\text{MFT}}) - L_S(h_{\text{FFT}})\right] \\ &+ \left[\Phi(C_{\text{MFT}}) - \Phi(C_{\text{FFT}})\right], \end{aligned} \tag{9}$$

revised as

$$U(h_{\text{MFT}}) - U(h_{\text{FFT}}) = \Delta_{\text{train}} + \Delta_{\text{complexity}}. \tag{10}$$

We numerically show $\Delta_{\text{train}} + \Delta_{\text{complexity}} < 0$. Details are provided in the Appendix.

Mask Fine-Tuning Loss Landscape. Loss surface visualization is commonly used to analyze the model optimization process, providing insights into

the model's status from a loss perspective. Thus, we visualize the MFT loss surface to illustrate its training dynamics. Fig. 5 shows the surface of LLaMA2-7B on math and instruction following domains. The Best FFT, continued FFT, and MFT are represented by the green star, the purple rectangle, and the red triangle, respectively. We find that MFT always optimizes the model to a better status than the Best FFT start point and its continued version in both training and test scenarios. We supplement more theoretical analyses in the Appendix.

4 Related Work

This work explores optimizing a well-trained large language model (LLM) by learning a binary mask and proposes mask fine-tuning (MFT), an upgrade to the current LLM fine-tuning protocol. We summarize the relevant literature on our work from (1) LLM pretraining and fine-tuning and (2) sparse network perspectives, emphasizing their correlations and differences.

4.1 LLMs Pre-training & Fine-tuning

LLM pre-training leads to significant performance gain for general language model capacity for both understanding (Devlin et al., 2019; Raffel et al., 2020; Liu, 2019; He et al., 2021) and generation (Radford et al., 2019; Brown et al., 2020; Achiam et al., 2023; Grattafiori et al., 2024; Touvron et al., 2023). Massive language corpora with large-scale pretraining enable them to serve as strong language backbones with basic common knowledge. Following pretraining, downstream fine-tuning on small but high-quality target datasets further enhances pre-trained models with the specific capacities of target domains (Hendrycks et al., 2021; Yu et al., 2024; Cobbe et al., 2021; Chen et al., 2021; Austin et al., 2021; Zhou et al., 2023; Wang et al., 2023; Nguyen, 2023). Different fine-tuning optimization strategies deliver various LLM properties, such as supervised fine-tuning (SFT), which follows a pre-training objective to involve extra domain knowledge into LLM (Taori et al., 2023; Chiang et al., 2023; Wang et al., 2022b), and policy-based fine-tuning methods better align the human preference with LLM (Schulman et al., 2017; Rafailov et al., 2024; Ivison et al., 2024; Ouyang et al., 2022; Dai et al., 2024). We propose a mask fine-tuning (MFT) as a post-fine-tuning strategy to further refine the fine-tuned LLM, especially for SFT in this study. Such a strategy can be flexibly deployed after any fine-tuned model to propose a new protocol to improve LLMs.

4.2 Sparse Networks

Model sparsity in neural networks is close to neural network pruning (Liu et al., 2019; Molchanov et al., 2017; Wang et al., 2022a) for model compression (Han et al., 2016b; Iandola, 2016) and acceleration (Han et al., 2016a; Wang et al., 2021; Wang and Fu, 2023; Ma et al., 2023; Fang et al., 2023). Our work overlaps with network pruning, as both aim to identify a model partition with a given sparsity. However, pruning aims to remove a large portion of the model for efficiency, whereas ours focuses on removing parameters that are irrelevant or even harmful to the model's capacity, which may require only a small model partition. More importantly, pruning methods inevitably entail a performance loss in the trained model, whereas ours focuses on further enhancing it. In addition, the sparsity concept is also involved in other machine learning fields such as network optimization (Srivastava et al., 2014; Srinivas et al., 2017; Baldi and Sadowski, 2013), model architecture design (Shazeer et al., 2017; Zoph, 2017; Elsken et al., 2019), lottery ticket hypothesis (Frankle and Carbin, 2019; Zhou et al., 2019; You et al., 2020; Bai et al., 2022b), neural network mechanism (Wortsman et al., 2020; Ramanujan et al., 2020; Bai et al., 2022a), etc. In our study, we introduce the concept of sparsity to investigate whether all parameters are beneficial for a well-trained LLM and to further improve the model by learning and applying a mask to it, proposing a new way for LLM fine-tuning.

5 Conclusion

In this paper, we challenge the necessity of structural integrity in large language models (LLMs) by examining whether a well-trained LLM can be further improved by removing certain model parameters. We propose mask fine-tuning (MFT) to explore it using a model with sufficient full fine-tuning (FFT), which consistently yields further improvements, generally enhances LLM capabilities, and naturally upgrades the LLM fine-tuning pipeline. Comprehensive experiments with detailed ablations support our conclusion, drawing on exploration intuitions, covering different pre-trained LLM backbones, downstream domains, and fine-tuning configurations. Meanwhile, MFT extends the functionality of model sparsity from compression to a broader capability scenario, inspiring further work.

Limitations

Given the vast number of pre-trained models, domains, and evaluation benchmarks in the community, this work considers only representative ones to provide a complete exploration pipeline. In addition, we consider only pure language models in this work, and extending it to the multi-modal domain will further support our statement and conclusion. Through our exploration, we observe interesting phenomena, including the sensitivity trend of MFT (Fig. 3), MFT showing promising results with very limited optimization steps (Fig. 1), initial try of global MFT not consistently working well (Fig. 6), etc. A detailed investigation of these points could further strengthen the MFT.

Ethics Statement

In our paper, we strictly adhere to the ARR ethical research standards and applicable laws. To the best of our knowledge, our work abides by the General Ethical Principles.

Reproducibility Statement

We adhere to ARR reproducibility standards and ensure the reproducibility of our work. All datasets we employed are publicly available. We will provide the code to reviewers and area chairs via an anonymous link, if needed, during the discussion period.

The Use of AI Assistant

We use the LLM only to correct grammar errors.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. 2018. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yue Bai, Huan Wang, Xu Ma, Yitian Zhang, Zhiqiang Tao, and Yun Fu. 2022a. Parameter-efficient masking networks. In *NeurIPS*.

Yue Bai, Huan Wang, Zhiqiang Tao, Kunpeng Li, and Yun Fu. 2022b. Dual lottery ticket hypothesis. In *ICLR*.

Pierre Baldi and Peter J Sadowski. 2013. Understanding dropout. In *NeurIPS*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *NeurIPS*.

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, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *https://vicuna.lmsys.org*.

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*.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2024. Safe rlhf: Safe reinforcement learning from human feedback. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. In *COLM*.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. In *JMLR*.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2023. Structural pruning for diffusion models. In *NeurIPS*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2024. Mathematical capabilities of chatgpt. In *NeurIPS*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. 2016a. Eie: Efficient inference engine on compressed deep neural network. In *ACM SIGARCH Computer Architecture News*.

Song Han, Huizi Mao, and William J Dally. 2016b. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *ICML*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Forrest N Iandola. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. In *NeurIPS*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations – democratizing large language model alignment. In *NeurIPS*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. In *COLM*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath tir.

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.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the value of network pruning. In *ICLR*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2024. Wizardcoder: Empowering code large language models with evol-instruct. In *ICLR*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*.

David A McAllester. 1998. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *ICLR*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Ha-Thanh Nguyen. 2023. A brief report on lawgpt 1.0: A virtual legal assistant based on gpt-3. *arXiv preprint arXiv:2302.05729*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *JMLR*.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. 2020. What's hidden in a randomly weighted neural network? In *CVPR*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. 2017. Training sparse neural networks. In *CVPR Workshops*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. In *JMLR*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

A Vaswani. 2017. Attention is all you need. In *NeurIPS*.

Guangyu Wang, Guoxing Yang, Zongxin Du, Longjun Fan, and Xiaohu Li. 2023. Clinicalgpt: large language models finetuned with diverse medical data and comprehensive evaluation. *arXiv preprint arXiv:2306.09968*.

Huan Wang and Yun Fu. 2023. Trainability preserving neural pruning. In *ICLR*.

Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. 2022a. Recent advances on neural network pruning at initialization. In *IJCAI*.

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. 2021. Neural pruning via growing regularization. In *ICLR*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022b. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. 2020. Supermasks in superposition. In *NeurIPS*.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. 2020. Drawing early-bird tickets: Towards more efficient training of deep networks. In *ICLR*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *ICLR*.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *NeurIPS*.

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

B Zoph. 2017. Neural architecture search with reinforcement learning. In *ICLR*.

Table 3: The information of training and test datasets used in our experiments.

Domain	Train / Test	Dataset Name	# of Samples
Math	Train	Tulu 3 Persona MATH (Lambert et al., 2025)	149,960
		Tulu 3 Persona GSM (Lambert et al., 2025)	49,980
		Tulu 3 Persona Algebra (Lambert et al., 2025)	20,000
		MetaMathQA (Yu et al., 2024)	395,000
		NuminaMath-TIR (LI et al., 2024)	64,312
	Test	GSM8K (Cobbe et al., 2021)	1,320
		MATH (Lewkowycz et al., 2022)	5,003
Coding	Train	Evol CodeAlpaca (Luo et al., 2024)	107,276
		Code-Alpaca (Chaudhary, 2023)	20,022
		Tulu 3 Persona Python (Lambert et al., 2025)	34,999
	Test	HumanEval (Chen et al., 2021)	164
		HumanEval+ (Liu et al., 2023)	164
Instruction Following	Train	OpenAssistant Guanaco (Köpf et al., 2023)	7,132
		Tulu 3 Persona IF (Lambert et al., 2025)	29,980
		Open-Orca (Mukherjee et al., 2023)	30,000
	Test	IFEval (Zhou et al., 2023)	100
		AlpacaEval 2.0 (Li et al., 2023)	805

A Implementation Details

We conduct our experiments on 8*A100 GPUs. To ensure that full fine-tuning (FFT) is sufficient, we set 4 as the maximum number of epochs for FFT, as we observe overfitting with a performance drop. We fine-tune MFT for 2 additional epochs, starting from the best FFT checkpoint (a total of 4 epochs). We set the batch size to 8, the accumulation steps to 16, and the weight decay to 0.0 for all settings. We use learning rates of 2e-5 and 2e-6 for LLaMA2-7B and LLaMA3.1-8B, with a linear schedule. The warm-up learning rate starts at 0 and increases linearly, with a warm-up ratio of 0.03 of the total training. We summarize details of the training and test dataset used for our experiments in Tab. 3.

B More Evaluations

Tab. 4 and Tab. 5 extend the performance tables in our main draft by adding cross-domain evaluations for our MFT, FFT, and other comparison methods, under domain-specific FFT settings. The best performance for each setting is highlighted in bold.

Table 4: Cross-domain evaluations on LLaMA2-7B under domain-specific FFT setting.

Method	Math		Coding		Instruction Following	
	GSM8K	Math	HumanEval	HumanEval+	IF-Eval	Alpaca-Eval
Pre-trained Model	15.2	2.5	25.8	22.4	34.3	0.5
Best FFT [Math]	46.9	6.7	17.7	15.9	29.3	1.5
Best FFT [Coding]	14.6	2.8	29.3	25.0	8.3	1.4
Best FFT [IF]	25.0	2.4	16.5	13.4	41.4	1.7
Continued FFT [Math]	45.2	5.7	19.5	17.1	33.2	1.5
Continued FFT [Coding]	11.1	2.5	28.0	23.8	13.1	2.7
Continued FFT [IF]	23.1	2.1	13.1	12.8	37.4	2.0
Random Mask w/ Best FFT [Math]	38.4	4.9	14.8	13.1	25.5	1.0
Random Mask w/ Best FFT [Coding]	12.4	2.5	28.0	24.0	7.5	1.1
Random Mask w/ Best FFT [IF]	23.0	2.3	14.4	12.2	39.4	1.5
L1 Mask w/ Best FFT [Math]	42.6	5.7	16.4	14.6	28.3	1.1
L1 Mask w/ Best FFT [Coding]	12.8	2.6	28.9	24.5	7.8	1.0
L1 Mask w/ Best FFT [IF]	23.9	2.4	14.9	12.7	40.9	1.5
[Ours] Best FFT + Mask [Math]	**47.3**	**7.6**	16.5	14.6	30.1	0.5
[Ours] Best FFT + Mask [Coding]	13.8	2.8	**31.7**	**28.0**	8.5	2.1
[Ours] Best FFT + Mask [IF]	23.5	2.8	15.2	11.6	**44.1**	**2.9**

Table 5: Cross-domain evaluations on LLaMA3.1-8B under domain-specific FFT setting.

Method	Math		Coding		Instruction Following	
	GSM8K	Math	HumanEval	HumanEval+	IF-Eval	Alpaca-Eval
Pre-trained Model	55.9	14.6	42.1	37.8	17.6	9.0
Best FFT [Math]	77.0	24.3	35.4	29.9	31.5	0.6
Best FFT [Coding]	60.7	13.8	51.2	45.1	5.3	0.9
Best FFT [IF]	63.0	10.1	50.0	44.5	59.8	12.0
Continued FFT [Math]	74.6	24.0	33.5	31.1	32.2	0.7
Continued FFT [Coding]	59.6	13.1	48.8	42.1	4.8	1.4
Continued FFT [IF]	63.6	8.7	45.7	43.5	57.2	11.4
Random Mask w/ Best FFT [Math]	74.9	22.3	34.3	28.5	30.0	0.4
Random Mask w/ Best FFT [Coding]	56.2	12.7	48.2	42.7	4.7	0.7
Random Mask w/ Best FFT [IF]	61.3	9.2	45.3	41.7	58.8	11.5
L1 Mask w/ Best FFT [Math]	75.1	22.7	34.4	30.6	31.1	0.6
L1 Mask w/ Best FFT [Coding]	57.3	12.9	49.1	44.1	4.8	0.8
L1 Mask w/ Best FFT [IF]	62.3	9.3	46.1	42.4	59.8	11.3
[Ours] Best FFT + Mask [Math]	**77.3**	**24.9**	36.0	32.2	31.1	0.5
[Ours] Best FFT + Mask [Coding]	58.6	13.6	**53.7**	**47.0**	4.9	0.0
[Ours] Best FFT + Mask [Instruction]	63.2	9.6	44.5	40.9	**65.8**	**13.8**

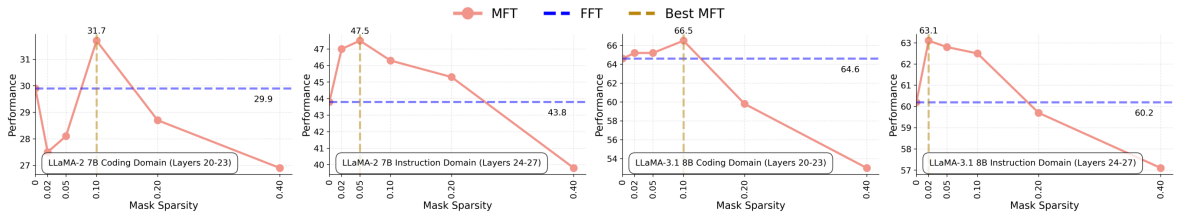

Figure 6: Masking ratio ablation visualizations. We use coding and instruction domains on LLaMA2 and LLaMA3.1. We observe that the original 10% ratio works well for coding but not for instruction, indicating that the masking ratio matters and that there is more MFT potential.

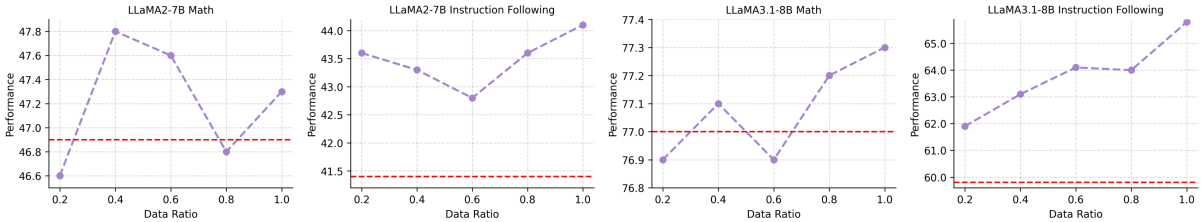

Figure 7: Data ratio ablation visualizations. We use math and instruction domains on LLaMA2 and LLaMA3.1. Compared with Best FFT (red dashed line), we observe that MFT (purple) consistently improves performance on the full dataset, but may still yield a promising performance gain with less data.

C More Ablations and Analyses

Masking Ratio Ablation. We ablate the masking ratio for local MFT. We use a masking ratio of 10% across all settings (Tab. 1 and Tab. 2), which may not be optimal. Here, we define a series of masking ratios for the ablation study. Specifically, we use all three domains on two backbones and set the masking ratio from 0.02 to 0.4. Results of coding and instruction are shown in Fig. 6, and the math results are in the supplementary. We find that our original 10% ratio works well for certain coding cases. However, it is not optimal for the instruction domain, where a smaller ratio works better than 10%. This observation indicates that the masking ratio of MFT affects the final performance.

Data Ratio Ablation. We ablate data ratios for MFT. Fig. 7 shows the performance trends of math and instruction following domains on LLaMA2-7B and LLaMA3.1-8B (others are in the supplementary). We find that MFT (purple) with the full dataset consistently improves the model compared with the Best FFT (red dashed line). However, less data may also perform competitively more efficiently.

Global Masking Fine-Tuning. Unlike the local MFT above, we initially explore MFT in a global setting. Instead of pre-defining the sparse ratio, we set a threshold (-0.035 here) for the weight score c_l in global MFT, retaining only weights with scores above the threshold. We expect it to encourage the MFT to automatically detect the appropriate sparse ratio. We use the math domain on LLaMA2-7B and LLaMA3.1-8B for the global MFT. The results in Tab. 6 show that the global MFT performs better than the local one using LLaMA2 on GSM8K, but causes a performance drop using LLaMA3.1-8B. Such results indicate that the global mask has greater potential for further improvement but may require more systematic investigation, and we leave this to future work.

Table 6: Initial exploration of global MFT on the math domain using LLaMA2-7B and LLaMA3.1-8B backbones. We observe better performance on LLaMA2-7B, especially for GSM8K, but a performance drop on LLaMA3.1-8B.

Method	LLaMA2-7B		LLaMA3.1-8B	
	GSM8K	Math	GSM8K	Math
Pre-Trained Model	15.2	2.5	55.9	14.6
Best FFT	46.9	6.7	77.0	24.3
Continued FFT	45.2 ↓ 1.7	5.7 ↓ 1.0	74.6 ↓ 2.4	24.0 ↓ 0.3
Global MFT w/ Best FFT **(Ours)**	49.0 ↑ 2.1	7.1 ↑ 0.4	74.1 ↓ 2.9	21.8 ↓ 2.5

Larger Layer-wise Group Ablation. The results of larger layer-wise group (16 layers) ablation for local MFT are shown in Fig. 8.

Fine-grained Evaluation Details. We provide fine-grained evaluation details for FFT and MFT on three domains using LLaMA2-7B in Tab. 8 (domain-specific FFT) and Tab. 9 (domain mixed-up FFT). They show the epoch number for Best FFT, Continued FFT, Best MFT, and Complete

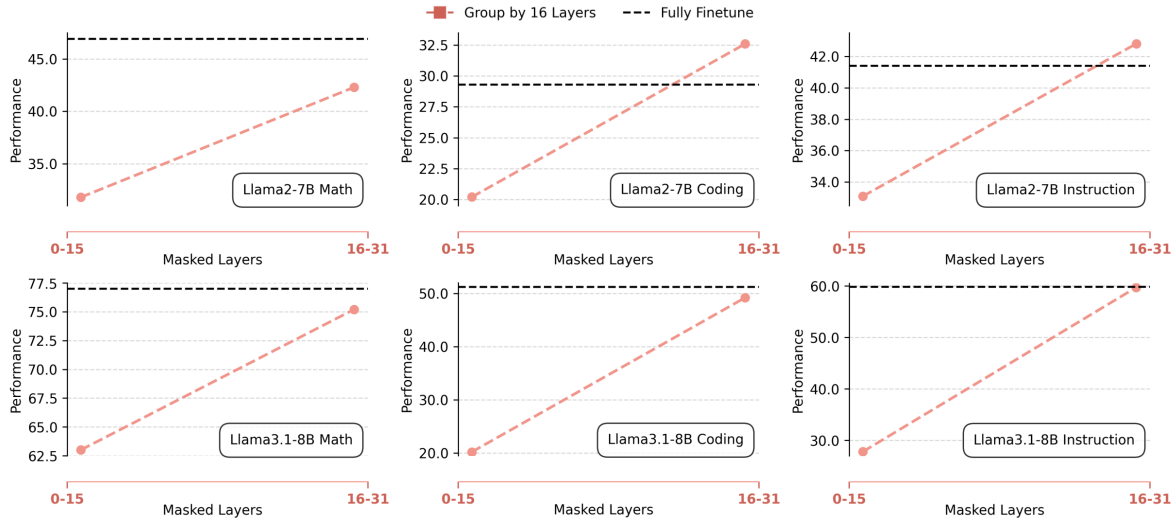

Figure 8: The local MFT ablation results of the larger layer-wise group (16-layer).

MFT, along with corresponding token usage information. The number of Continued FFT is consistent at 4, and those for Best FFT are less than 4. Based on the Best FFT, we then perform 2 epochs of MFT. Therefore, the number of Complete MFT is always equal to the number of Best FFT plus 2, but Best MFT may be less than it. For example, for the math domain in Tab. 8, Continual FFT is 4 epochs, while FFT achieves the best results at 2.4 epochs as Best FFT. Starting from 2.4 epochs, we perform MFT for an additional 2 epochs. Thus, the Complete MFT is 2.4 + 2 = 4.4 epochs, while the Best MFT is 4 epochs, indicating that MFT uses 4 - 2.4 = 1.6 epochs to achieve the best result based on Best FFT. For each setting, the number of tokens used is provided in brackets.

More Details of Theoretical Analysis. For our case, to theoretically support that our MFT has better optimization potential than FFT, we need to verify that $\Delta_{\text{train}} + \Delta_{\text{complexity}} < 0$ in Eq. 10.

For the first term, we first supplement a set of exemplar training loss statistics in Tab. 7. It shows the mean training loss after training stabilizes, indicating that MFT can further reduce the training loss compared with the model after FFT (the Best FFT) on which it is based.

Table 7: Exemplar training loss statistics of LLaMA2-7B models after training is stable.

LLaMA2-7B	Math	Coding	Instruction
FFT loss	0.101	0.098	0.125
MFT loss	0.085	0.054	0.060

Then, we can ensure the first term $\Delta_{\text{train}} < 0$.

For the second term, we need to compare the encoded model complexity. Given d as the total number of weights, z as the weights being masked out (as zero), and b as the bit width. Since we are using weight-level masking granularity, we have

$$C(h_{\text{FFT}}) \approx bd, \quad C(h_{\text{MFT}}) \approx b(1-p)d + \log_2 \binom{d}{z}, \tag{11}$$

where $p = z/d$ represents the sparse ratio in our case. The first term of C_{MFT} is the encoding cost of non-zero weights, and the second is for the index cost of zero weights. Further, we have $\log_2 \binom{d}{z} \approx dH(p)$, where

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p). \tag{12}$$

Then, the second term can be described as

$$\Delta_{\text{complexity}} = C(h_{\text{MFT}}) - C(h_{\text{FFT}}) \approx d[H(p) - bp]. \tag{13}$$

For practice, we always use $b = 8$ or 16. We use a 10% sparse ratio for our MFT, applied to a 4-layer group, yielding a global sparse ratio of $2.7\% \sim 3\%$ for two backbones. To consider the strict case, we use $b = 8$ and a ratio of 2.7% in the calculation shown below.

$$H(p) - bp = H(0.027) - 8 \cdot 0.027 \approx -0.037 < 0. \tag{14}$$

Then, we have the second term $\Delta_{\text{complexity}} < 0$.

Combining both the first and second terms, we have

$$U(h_{\text{MFT}}) - U(h_{\text{FFT}}) = \Delta_{\text{train}} + \Delta_{\text{complexity}} < 0. \tag{15}$$

Table 8: Training epochs and corresponding number of used tokens (in brackets) on LLaMA2-7B under domain-specific FFT setting.

Domain	Best FFT	Continued FFT	Best MFT	Complete MFT
Math	2.4 epochs (5.90e8)	4.0 epochs (9.83e8)	4.0 epochs (9.83e8)	4.4 epochs (1.08e9)
Coding	2.7 epochs (3.32e8)	4.0 epochs (4.92e8)	3.6 epochs (4.43e8)	4.7 epochs (5.78e8)
Instruction Following	2.4 epochs (5.51e8)	4.0 epochs (9.18e8)	3.2 epochs (7.35e8)	4.4 epochs (1.01e9)

Table 9: Training epochs and corresponding number of used tokens (in brackets) on LLaMA2-7B under mixed-up FFT setting.

Domain	Best FFT	Continued FFT	Best MFT	Complete MFT
Math	2.0 epochs (2.67e9)	4.0 epochs (5.34e9)	3.7 epochs (3.09e9)	4.0 epochs (3.16e9)
Coding	2.0 epochs (2.67e9)	4.0 epochs (5.34e9)	3.3 epochs (2.83e9)	4.0 epochs (2.92e9)
Instruction Following	2.0 epochs (2.67e9)	4.0 epochs (5.34e9)	4.0 epochs (3.13e9)	4.0 epochs (3.13e9)

Finally, we conclude that $U(h_{\mathrm{MFT}}) < U(h_{\mathrm{FFT}})$, which means the MFT can further reduce the test loss upper bound compared with FFT and theoretically support our proposed method.

More Training Cost Comparisons. We supplement the training cost comparisons with GPU memory usage, token count, and training time for the math and instruction-following domains in Fig. 9.

More Mask Ratio Ablation. The mask ratio ablation on the math domain using LLaMA2-7B and LLaMA3.1-8B is shown in Fig.10.

More Data Ratio Ablation. The MFT data ratio ablation on coding domain using LLaMA2-7B and LLaMA3.1-8B is shown in Fig. 11.

More Details of Loss Landscape Visualization. A loss landscape visualization in the coding domain using LLaMA2-7B is shown in Fig. 12.

For the loss surface analysis, we present empirical results as loss-landscape visualizations in Fig. 5 and Fig. 12. Here, we supplement the theoretical view with further support.

We assume the model after FFT (Best FFT) has Hessian as

$$H_{\mathrm{FFT}} = \nabla^2 L(\theta_{\mathrm{FFT}}) \succeq 0. \tag{16}$$

Since MFT learns a mask and applies it to the model after FFT, which means the MFT model is a projection P of the FFT model, where the corresponding Hessian can be given by

$$H_{\mathrm{MFT}} = P H_{\mathrm{FFT}} P. \tag{17}$$

Since for $H_{\mathrm{FFT}} \succeq 0$ and its any projection (H_{MFT} for here), we have

$$\begin{aligned} \lambda_{\max}(H_{\mathrm{MFT}}) &\leq \lambda_{\max}(H_{\mathrm{FFT}}), \\ \mathrm{Tr}(H_{\mathrm{MFT}}) &\leq \mathrm{Tr}(H_{\mathrm{FFT}}). \end{aligned} \tag{18}$$

As $\lambda_{\max}$ and $\mathrm{Tr}(\cdot)$ represent the model sharpness and mean curvature, lower values of them indicate the model has a flatter status, which means the MFT model has better generalization property than the FFT model. Such analysis provides theoretical support and aligns with the empirical results shown in the landscape visualizations (the MFT model is visually flatter than the FFT model).

D Mask Visulization

In Fig. 13 and Fig. 14, we visualize the head-level mask sparsity patterns learned on LLaMA2-7B of the coding domain under 10% sparsity, using six different random seeds. The patterns across all six seeds are remarkably consistent. This indicates that MFT performs fine-grained, weight-level subnetwork selection that subtly reweights specific attention directions. The convergence of zero-value locations across seeds further demonstrates that, for the same model and domain, MFT learns a structured, stable, and reproducible subnetwork rather than random noise. These results align with our design intuition that MFT identifies domain-relevant structural preferences with high consistency.

E More Discussion

Intuition. MFT is highly relevant to a range of well-established research topics, such as sparse network training and network pruning. However, inspired by their techniques, we approach sparsity from a novel perspective, shifting from typical model efficiency to a general capability perspective by removing negative components to improve performance. In other words, sparse models typically aim to compress the model while maintaining

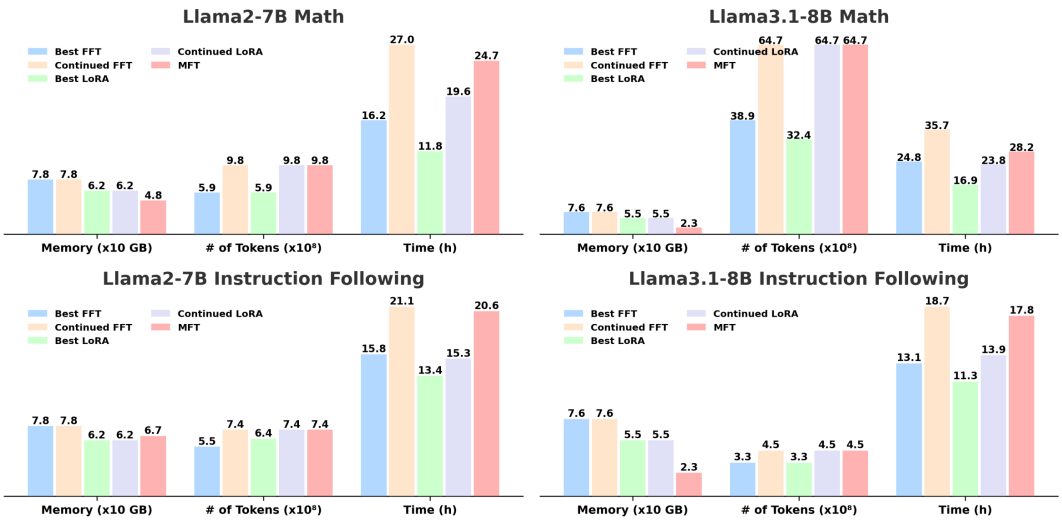

Figure 9: Training cost comparisons of GPU memory, used token number, and training time on math and instruction following domain using LLaMA2-7B and LLaMA3.1-8B.

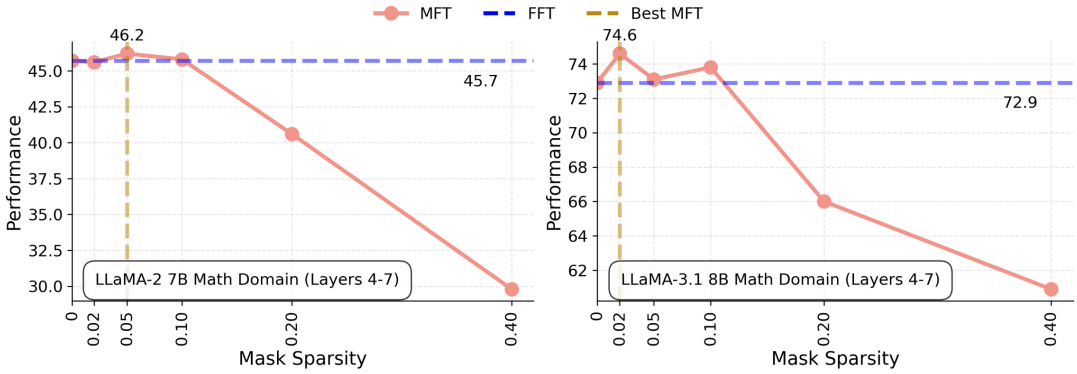

Figure 10: Masking ratio ablation study on the math domain using LLaMA2-7B and LLaMA3.1-8B.

performance (seen as *subtraction*). However, we use sparsity as a tool for model enhancement (seen as *addition*). We expect that such counterthinking about sparse networks could inspire broader explorations in the future to enrich their research potential.

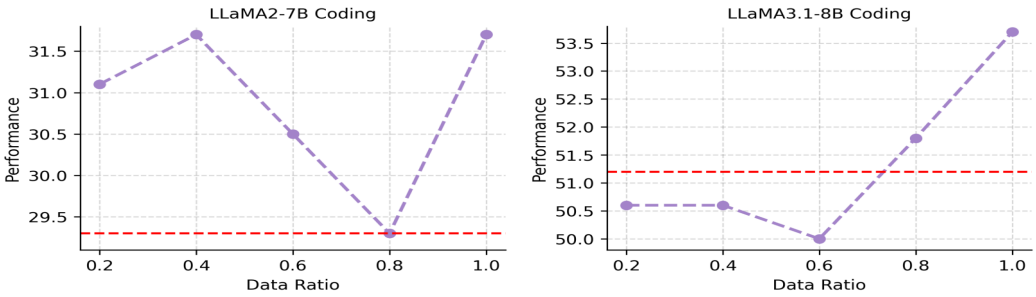

Figure 11: Data ratio ablation study on coding domain using LLaMA2-7B and LLaMA3.1-8B.

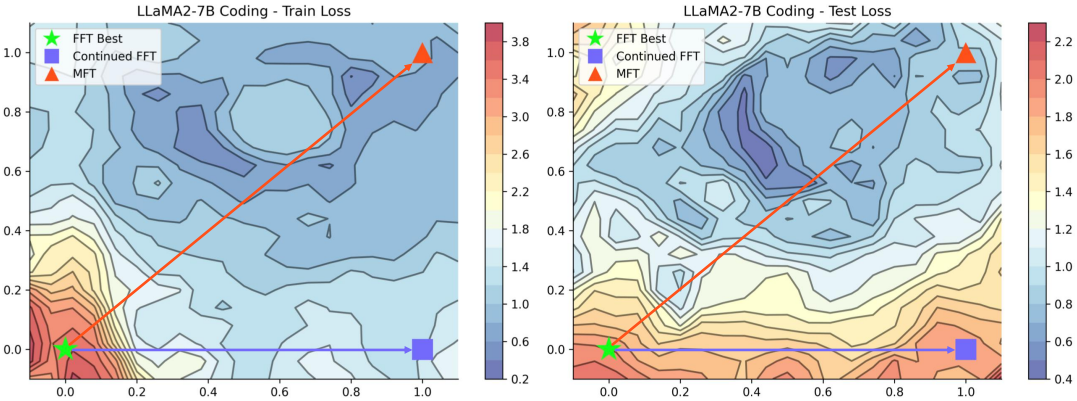

Figure 12: The loss landscape visualization on the coding domain using LLaMA2-7B.

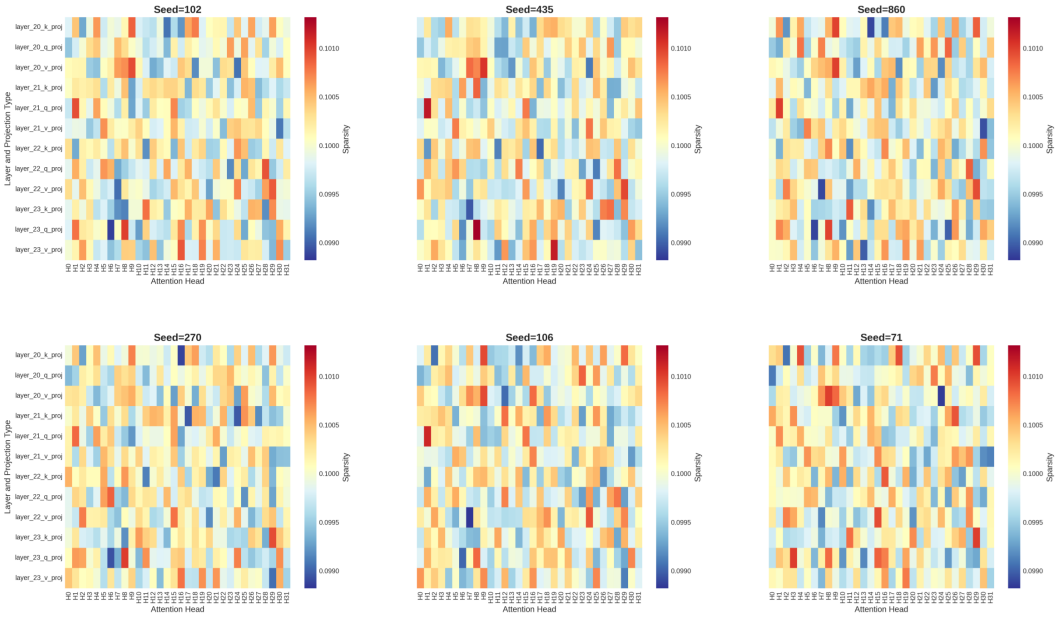

Figure 13: Head-level mask sparsity distributions learned on LLaMA2-7B of the coding domain under 10% sparsity, using six different random seeds. Each heatmap shows the average sparsity values of Q/K/V projections across attention heads for layers 20–23.

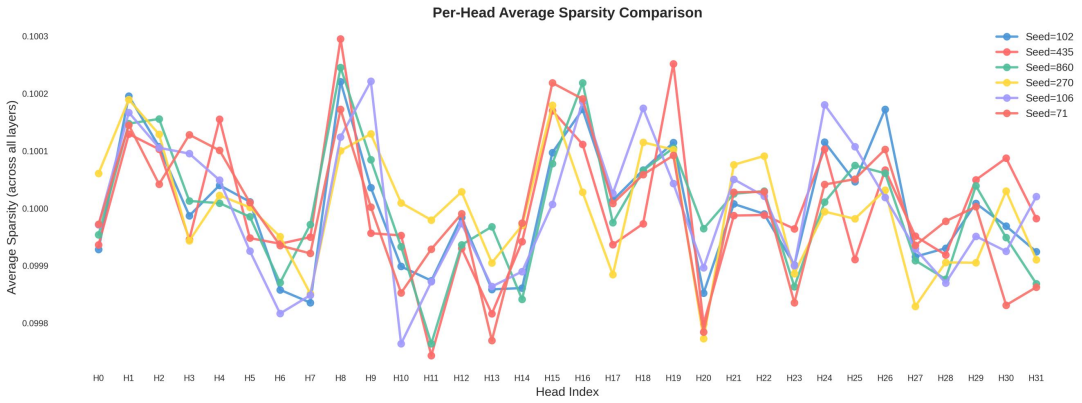

Figure 14: The visualization of average sparsity across layers of each head for six different seeds under the same 10% sparsity configuration on LLaMA2-7B of coding domain (layers 20–23). All six curves exhibit highly similar fluctuation patterns across the 32 heads, showing that MFT consistently learns the same attention directions across seeds.