Balancing Speciality and Versatility: a Coarse to Fine Framework for Supervised Fine-tuning Large Language Model

Hengyuan Zhang[1†], Yanru Wu[1†], Dawei Li[3], Sak Yang[4], Rui Zhao[2], Yong Jiang[1*], Fei Tan[2*$]

[1]Tsinghua University [2]SenseTime Research
[3]University of California, San Diego [4]Independent Researcher
{zhang-hy22,wu-yr21}@mails.tsinghua.edu.cn dal034@ucsd.edu
sakyang@outlook.com {zhaorui,tanfei}@sensetime.com

Abstract

Aligned Large Language Models (LLMs) showcase remarkable *versatility*, capable of handling diverse real-world tasks. Meanwhile, aligned LLMs are also expected to exhibit *speciality*, excelling in specific applications. However, fine-tuning with extra data, a common practice to gain speciality, often leads to catastrophic forgetting (CF) of previously acquired versatility, hindering the model's performance across diverse tasks. In response to this challenge, we propose *CoFiTune*, a coarse to fine framework in an attempt to strike the balance between speciality and versatility. At the coarse-grained level, an empirical tree-search algorithm is utilized to pinpoint and update specific modules that are crucial for speciality, while keeping other parameters frozen; at the fine-grained level, a soft-masking mechanism regulates the update to the LLMs, mitigating the CF issue without compromising speciality. In an overall evaluation of both speciality and versatility, *CoFiTune* consistently outperforms baseline methods across diverse tasks and model scales. When compared to the full-parameter SFT, *CoFiTune* offers an average versatility improvement of 14%, while only incurring a marginal loss in speciality. Lastly, based on further analysis, we provide a speculative insight into the information forwarding process in LLMs, which helps explain the effectiveness of the proposed method. The code is available at https://github.com/rattlesnakey/CoFiTune.

1 Introduction

Aligned LLMs mainly undergo a two-step procedure: initial pre-training on web-scale text corpora, followed by fine-tuning on diverse instructions to align with human intentions. They exhibit remarkable **versatility**, *showcasing their ability to handle various real-world tasks, such as reasoning, common sense question-answering, and instruction following* (Zhao et al., 2023; Achiam et al., 2023; Lu et al., 2023b).

Figure 1: An illustration of our objective: achieving effective speciality without significantly compromising versatility.

Despite the versatility, aligned LLMs still *fall short in certain tasks or domains, such as mathematics* (Gou et al., 2023), *finance* (Li et al., 2023c), *and law* (Cui et al., 2023a). To bolster performance in these particular tasks or domains, i.e., to gain **speciality**, a common practice is fine-tuning. However, during the fine-tuning process, the modification of model parameters often leads to catastrophic forgetting (CF), thereby causing a noticeable loss of versatility (Lin et al., 2023b). This loss adversely affects the performance of fine-tuned models across various real-world tasks (Cheng et al., 2023; Dong et al., 2023), propelling several works to investigate and contribute solutions to the CF in LLM versatility (Lin et al., 2023b; Wang et al., 2023c).

As a relatively new problem, the CF in LLMs remains under-explored. We categorize the existing studies into regularization-based, weight-based, and architecture-based methods[1]. Regularization-based methods (Lin et al., 2023b; Smith et al., 2023) add extra terms into loss function to penalize parameter changes. Weight-based methods (Wortsman et al., 2022; Ke et al., 2022) introduce weight coefficients for parameters to regulate their updates. Architecture-based methods (Wang et al., 2023d; Razdaibiedina et al., 2023) design and exclusively

†Work was done during internship at SenseTime Research
* Corresponding author
$ Project lead

[1]See Appendix A.5 for more details about these methods.

Findings of the Association for Computational Linguistics ACL 2024, pages 7467–7509
August 11-16, 2024 ©2024 Association for Computational Linguistics

fine-tune extra modules outside the original model. However, these methods still have limitations: 1) Regularization- and weight-based methods involve fine-tuning all parameters, which poses a significant challenge to preventing CF in versatility. Particullary, CF deteriorates as the training iteration proceeds (Luo et al., 2023b). 2) Architecture-based methods only update the inserted parameters, which blocks the learning of speciality, particularly on challenging tasks (Lin et al., 2023b).

These limitations raise a key question: *How can the model gain speciality while preserving the versatility, thereby enhancing its overall abilities?* (shown in Fig. 1). Recent research threads highlight two pivotal findings: the presence of redundant parameters in the model (Bansal et al., 2023; Sun et al., 2023) and the distinct role played by different modules and layers (Geva et al., 2021; Meng et al., 2022a). These suggest that it is highly feasible to enhance speciality by updating specific parameters within a defined layer range while keeping the remainder frozen to maintain versatility.[2]

Therefore, we propose a **Co**arse to **Fi**ne framework, i.e., *CoFiTune*. At the coarse-grained level, we perform an empirical tree-search algorithm to identify and update specific modules within a defined layer range that are crucial for speciality without significant versatility penalty. Simultaneously, the remaining parameters are frozen to further preserve versatility. Subsequently, recognizing that not all units[3] in a module are evenly important for the versatility (Michel et al., 2019), we utilize a fine-grained soft-masking mechanism to regulate the backward gradient flow based on their importance values for versatility. This further mitigates the CF issue while not jeopardizing the speciality. To summarize, our contributions are as follows:

1) We present *CoFiTune*, a framework striking a delicate balance between versatility and speciality. We also lead in creating a comprehensive Chinese CF setting, contributing to CF research in the Chinese domain. Extensive experiments demonstrate the effectiveness of *CoFiTune* across diverse tasks and model scales.

2) Our *CoFiTune* achieves over 95% versatility and 90% speciality compared to original and full SFT models on average, and we find that only tuning the FFN module in the mid-to-lower layer range achieves satisfactory speciality without significantly affecting versatility.

3) We conduct extra experiments to give more insights, demonstrating the module's importance for gaining speciality and exploring the function of three crucial areas within the model.

2 Related Work

CF in LLM The previous studies about the CF problem typically focused on the CF of prior training tasks in a continual learning context, where a sequence of tasks are learned (Madotto et al., 2021; Wu et al., 2021). Recently, attention has shifted towards investigating the CF problem in the general abilities of LLMs. Luo et al. (2023b) first explored the CF problem in LLMs, revealing a pronounced CF phenomenon in general knowledge and reasoning. Subsequently, Wang et al. (2023c) established a benchmark comprising a sequence of challenging tasks to facilitate CF research in LLMs. Different from them, Lin et al. (2023b) proposed several solutions to address the trade-off between versatility and speciality in LLMs under a single task fine-tuning setting. In this study, we follow Lin et al. (2023b) to strike a balance between the speciality and versatility of LLMs.

Key Components in Transformer Previous works demonstrate the presence of substantial redundant parameters in Transformer-based models (An et al., 2020; Xia et al., 2022; Bansal et al., 2023). Therefore, it's crucial to pinpoint the key components and precisely comprehend their underlying mechanisms. Recent endeavors try to analyze the involved layers and modules. Elhage et al. (2021) developed mathematical tools to unveil an information-copying behavior within the attention module. Geva et al. (2021) analyzed the feed-forward network (FFN), considering the up and down projection matrices as the key and value of the memories. Mirzadeh et al. (2024) demonstrated that SwiGLU, a common trainable activation function in the FFN of LLMs, can be replaced with non-trainable ReLU to reduce computation without affecting performance. In light of the aforementioned insights, in this work, we explore the key components, i.e., attention and FFN[4] modules in certain layer range that are integral to gaining speciality and preserving versatility.

[2]The results in Sec. 4.3 and 4.4, along with the insights 1, 3, and 4, further support our speculation.

[3]*Units* denote the attn. heads in MHA or neurons in FFN.

[4]For simplicity, starting from this section, FFN only refers to the combination of up & down projection.

3 The Framework

We start with the task formulation (Sec. 3.1) and the backbone we used in this work (Sec. 3.2). Then, we outline our *CoFiTune* framework, covering: 1) an empirical tree-search algorithm to identify modules within a defined layer range that balances speciality and versatility effectively (Sec. 3.3); 2) a Fine-SoftMask mechanism regulating the backward gradient flow based on units' importance for versatility to further mitigate the CF issue (Sec. 3.4).

3.1 Task Formulation

Different from earlier researches, which focus on the CF in previously learned abilities when fine-tuning on a sequence of tasks (Liu et al., 2021b; Zhang et al., 2022b), we align with Lin et al. (2023b), emphasizing the trade-off between speciality and versatility in fine-tuning original LLM for a single task. This is because the original versatility of LLM is crucial and undergoes a progressive reduction at each fine-tuning step (Luo et al., 2023b). Additionally, we investigate a potential strategy to replace sequential fine-tuning for better performance. Further discussion of our setting is in Appendix A.4.

During supervised fine-tuning (SFT), given an input token sequence $\boldsymbol{x} = (x_0, x_1, \ldots)$, the model is trained to predict the next token x_i in an autoregressive manner:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{SFT}}} \left[-\sum_i \log p(x_i | x_{<i}; \theta) \right] \quad (1)$$

where θ represents the parameters of aligned LLM, $\mathcal{D}_{\text{SFT}}$ is the fine-tuning dataset.

After obtaining the fine-tuned model $\hat{\theta}$, we assess its speciality (Spec.) and versatility (Vers.) scores. A unified (Uni.) score is further defined to evaluate its overall ability, i.e., both Spec. and Vers.. Refer to Sec. 4.2 for more details of scores.

3.2 Backbone Architecture

In this work, we utilize the Llama-based (Touvron et al., 2023) aligned LLM as our backbone, which consists of an Embedding module, an LM Head, and a stack of Llama Layers. Each Llama Layer integrates a multi-head attention (MHA), followed by a feed-forward network (FFN) with residual connections. For an input $\mathbf{x}$, the Llama Layer generates output $\mathbf{y}$ based on the following equations:

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} + \text{MHA}(\text{Norm}(\mathbf{x})) \\ \mathbf{y} &= \mathbf{x}' + \text{FFN}(\text{Norm}(\mathbf{x}')) \end{aligned} \quad (2)$$

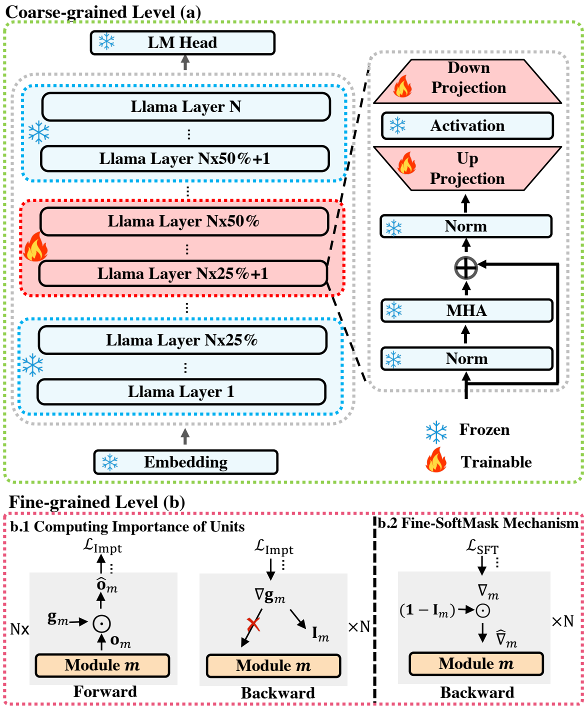

Figure 2: An illustration of our *CoFiTune* framework. N denotes the number of layers. At the coarse-grained level, we pinpoint the module (e.g., FFN) within a defined layer range (e.g., 10th - 20th layers) that gains speciality effectively without harming versatility much. At the fine-grained level, we selectively update the parameters within the region identified at the coarse-grained level and leverage $\mathbf{I}_m$ to control their gradient flow.

Each head in the MHA module can be defined as:

$$\text{head}_i = \text{Attention}(\mathbf{x}\mathbf{W}_i^Q, \mathbf{x}\mathbf{W}_i^K, \mathbf{x}\mathbf{W}_i^V) \quad (3)$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$ are the weight matrices of query, key, and value for the i-th head. The FFN module is parameterized by an up projection (up_proj) and a down projection (down_proj):

$$\text{FFN}(\mathbf{x}') = \sigma(\mathbf{x}'\mathbf{W}^1)\mathbf{W}^2 \quad (4)$$

where σ is the SwiGLU activation function (Shazeer, 2020), $\mathbf{W}^1$ and $\mathbf{W}^2$ are up_proj and down_proj respectively. More details of backbone information are illustrated in Appendix A.1.

3.3 Coarse-grained Level

Following the analysis in paragraph 2 of Sec. 2, which reveals the presence of parameter redundancy and key components (MHA and FFN) in LLMs, we aim to pinpoint the module (e.g., FFN) within a defined layer range (e.g., 10th - 20th layers) target for speciality without significantly affecting versatility. Subsequently, we exclusively train the identified module within the layer range,

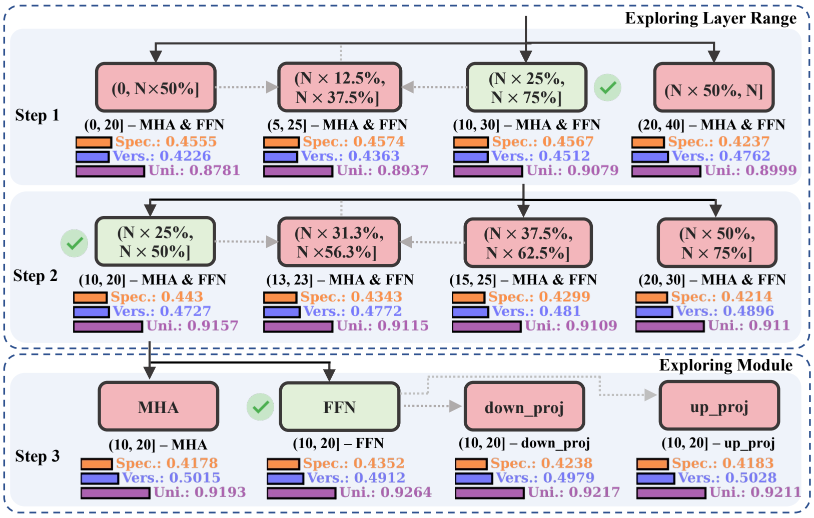

Figure 3: The exploration process of Finance task in the 13B model. N denotes the number of layers and in this case, $N = 40$. For simplicity, we denote the model fine-tuned in our exploration as "layer range - module", e.g., the model fine-tuned with FFN module within the layer range $(10, 20]$ denoted as "$(10, 20]$ - FFN".

while freezing the remaining parameters to further preserve the versatility.

Our exploration process encompasses three steps: 1) identifying the optimal broad layer range lr_broad; 2) narrowing down from lr_broad to obtain the optimal narrow layer range lr_narrow; 3) exploring the best modules within the narrow layer range lr_narrow. The best solution is up to the best Uni. score through three steps. Note that in steps 1 and 2, all modules, including both MHA and FFN are trainable in the specific exploring layer range; in step 3, only the selective module is trainable.[5]

Take the exploration process for Finance task in the 13B model as an example (illustrated in Fig. 3)[6]. In step 1, we first obtain the Uni. scores of three broad layer ranges: $(0, 20]$, $(20, 40]$, and $(10, 30]$. For a more accurate search, we further obtain the Uni. score of the layer range $(5, 25]$ between $(10, 30]$ with best Uni. score and $(0, 20]$ with second-best Uni. score. In step 2, we first split the optimal broad layer range lr_broad $(10, 30]$ into three narrow layer ranges: $(10, 20]$, $(20, 30]$, $(15, 25]$, and follow the exploration procedures from step 1 to identify the optimal narrow layer range lr_narrow $(10, 20]$.

In step 3, we separately explore the Uni. scores for the modules within the layer range $(10, 20]$, i.e., "$(10, 20]$ – MHA" and "$(10, 20]$ – FFN" and obtain the best module "$(10, 20]$ – FFN". To delve deeper, we further examine the Uni. scores of submodules (up_proj and down_proj) within the best module (FFN). Here, we opt not to explore smaller layer ranges within lr_narrow (e.g., $(10, 15]$ and $(15, 20]$) due to a substantial speciality gap compared to the all-layer range. See Appendix A.2 for the results of smaller layer ranges.

We select the best Uni. score among three steps as the final solution of the coarse-grained level. The entire exploration is illustrated in Algorithm 1.

3.4 Fine-grained Level

Recognizing that not all units (attention heads or neurons) in a module contribute equally to the LLM versatility (Michel et al., 2019), 1) we first compute their importance for LLM versatility; 2) Subsequently, we utilize a fine-grained soft-masking (Fine-SoftMask) mechanism to control the backward gradient flow based on their importance values, aiming to further mitigate the CF issue.

In this section, we use "module" or m to denote the module from the final solution identified at the coarse-grained level. Note that the Fine-SoftMask is exclusively applied to these modules, e.g., the FFN modules within the layer range (10, 20].

Computing Importance of Units Before directly fine-tuning the module, we follow Ke et al. (2022) to use a proxy based on robustness, i.e., KL-divergence loss, to compute units' importance for versatility without accessing the external data. This is achieved by employing a gradient-based importance detection method (Michel et al., 2019):

[5]We exclusively train the up_proj $\mathbf{W}^1$ and down_proj $\mathbf{W}^2$ within the FFN (shown in Eq. 4) and head's query $\mathbf{W}_i^Q$, key $\mathbf{W}_i^K$, value $\mathbf{W}_i^V$ in the MHA (shown in Eq. 3).

[6]The details of tasks and models are illustrated in Sec. 4.1.

$$\hat{\mathbf{o}}_m = \mathbf{g}_m \odot \mathbf{o}_m$$

$$\mathbf{I}_m = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{\partial \mathcal{L}_{\text{Impt}}}{\partial \mathbf{g}_m}\right| \tag{5}$$

$$\mathcal{L}_{\text{Impt}} = \text{KL}(f^1_{\text{LLM}}(\mathbf{x}), f^2_{\text{LLM}}(\mathbf{x}))$$

where $\mathbf{o}_m$ refers to the output of module m. K is the data size of $\mathcal{D}_{\text{SFT}}$. $\mathbf{g}_m$, an all 1's *virtual parameter*, remains unchanged during the computing process, as we only need its gradient $\nabla \mathbf{g}_m$ (the term within || in Eq.5) on each parameter to get the importance of corresponding unit. $\odot$ denotes element-wise multiplication, i.e., each $g_{m,i}$ in $\mathbf{g}_m$ corresponding to a unit in the module. The overall importance of units $\mathbf{I}_m$ is computed by averaging the gradient values of $\mathbf{g}_m$, where $\mathbf{I}_m$ is of the same size as $\mathbf{g}_m$.

f^1_{LLM} and f^2_{LLM} are the LLM with different dropout masks[7]. By simply feeding the same $\mathbf{x} \in \mathcal{D}_{\text{SFT}}$ to the model with different dropout masks twice, we obtain two representations. The difference between these two representations, computed by the KL-divergence loss, can measure the unit's importance for the model. The rationale is as follows: if an unimportant neuron is dropped out, the resulting representation will show minimal alteration. Therefore, we will obtain a "normal type" representation similar to the one without any dropout. Conversely, if an important neuron is dropped out, the resulting representation will significantly change, resulting in an "abnormal type" representation (since dropout simulates noise addition). The difference between abnormal and normal output representations will be considerable. By utilizing KL divergence loss, we can effectively capture this difference and assign high gradient values to determine the importance of neurons. In contrast, the difference between normal representations will be marginal, resulting in small gradient values and, consequently, small importance values.

Fine-SoftMask Mechanism Then, we regulate the backward gradient flow based on the computed $\mathbf{I}_m$. We initially derive the original gradient ∇_m by employing $\mathcal{L}_{\text{SFT}}$ loss defined in Eq. 1. Subsequently, we apply the $\mathbf{I}_m$ to obtain the modified gradient $\hat{\nabla}_m$ for updating:

$$\hat{\nabla}_m = (1 - \mathbf{I}_m) \odot \nabla_m \tag{6}$$

Here, we expand (by copying) the $\mathbf{I}_m$ to match the dimensions of ∇_m to apply it to all associated parameters. This mechanism further mitigates the CF issue by regulating the update intensity of parameters based on $\mathbf{I}_m$ for versatility. Note that the Fine-SoftMask is only applied in the backward pass during fine-tuning and the virtual parameter $\mathbf{g}_m$ mentioned in the computing stage will be discarded in this process. The overview of our *CoFiTune* is shown in Fig. 2.

4 Experiment

In this section, we conduct extensive experiments to answer the following questions: **1) RQ1**: Training which layers and modules can optimize the trade-off between speciality and versatility? **2) RQ2**: How does our *CoFiTune* framework compared to the baseline methods? **3) RQ3**: Can the Fine-SoftMask mechanism further alleviate the CF issue in versatility? **4) RQ4**: Can our *CoFiTune* be seamlessly applied to models across different model scales and families?

We also conduct further experimental analysis to delve into more insights for this field in Sec. 4.4.

4.1 Datasets and Experiment Settings

Datasets: Building upon our task formulation in Sec. 3.1, we establish a Chinese CF setting, aiming to advance research in the Chinese language. This fills a crucial gap in existing studies, which have predominantly focused on English (Razdaibiedina et al., 2023; Lin et al., 2023b; Luo et al., 2023b).

Our setting emphasizes both speciality and versatility. Specifically, for LLM's speciality, we select tasks with considerable complexity in two primary categories: **1) Knowledge Intensive Tasks**: These tasks require the model to generate responses to questions tailored for specific domain. We create the **Finance** and **Law** instruction format datasets based on FiQA (Wang et al., 2023b) and LlamaLawyer (Huang et al., 2023); **2) Ability Intensive Tasks**: These tasks aim to test LLM's ability to handle challenging and crucial tasks in real-world scenarios. We create the mathematical problem solving (**Math**) and Context-Aware Generation (**CAG**[8]) based on MGSM8k (Chen et al., 2023a) and DuReader (He et al., 2018).

To evaluate the versatility of LLM, we combine the insights from previous studies (Lin et al., 2023b;

[7] As the Llama layer lacks dropout module, we manually add it after the output of down_proj and the attention weights of MHA when computing the importance of units.

[8] CAG task requires the model to generate answers flexibly based on the given context (See Sec. A.6.2 for more details).

Luo et al., 2023b) and focus on three key aspects: **1) General Domain Knowledge (Gen-Kn.)**: We adopt CMMLU (Li et al., 2023b) to assess the original world knowledge stored in LLM; **2) Generic Reasoning (Gen-Rs.)**: We utilize commonly used reasoning datasets, including LogiQA (Liu et al., 2021a), LogiQA2 (Liu et al., 2023), OCNLI (Hu et al., 2020), and Zh-Winograd (Muennighoff et al., 2022); **3) Instruction Following (Instruct.)**: Following Lin et al. (2023b), we employ a Chinese GPT4-instruct dataset (Peng et al., 2023). Both Gen-Kn. and Gen-Rs. are evaluated in a zero-shot manner. Statistics and detailed descriptions of each dataset are provided in Appendix A.6.

Experimental Setting: We use the aligned LLM Chinese-Alpaca-Pro (Cui et al., 2023b) due to its robust versatility across multiple model scales, ranging from 7B to 33B, as detailed in Table 9. Implementation details, including learning rates and batch sizes tailored to different model scales and tasks, can be found in Appendix A.7. Fine-tuning adheres to the instruction prompt provided by Cui et al. (2023b), detailed in Appendix A.9.1.

4.2 Baselines and Metrics

Baselines: We compared our *CoFiTune* with full parameter SFT (Full SFT) and five CF baselines, with detailed descriptions provided in Appendix A.5. These baselines are carefully categorized into three groups:

- **Regularization-based** These methods introduce additional terms into the loss function to constrain changes in model weights. Selected baselines include L1 (Panigrahi et al., 2023) and L2 regularization (Lin et al., 2023b);
- **Weight-based** These methods design weight coefficients for parameters to control their updates. Selected baselines include Wise-FT (Wortsman et al., 2021) and V-SoftMask (Ke et al., 2022);
- **Architecture-based** These methods fine-tune only the external module, leaving the rest of the parameters fixed. Selected baseline includes LoRA (Hu et al., 2021).

Metrics: The exploration algorithm in Sec. 3.3 requires numerous experiments, demanding a fast, cost-effective, and accurate automatic evaluation strategy. For Finance, Law, and CAG, we employ automatic generation metrics encompassing both semantic alignment and n-gram matching, including BERTScore (Zhang et al., 2020), Rouge (Lin, 2004), and BLEU (Papineni et al., 2002). Additional experiments confirm a strong correlation between the automatic scores and those generated by GPT-4, as well as human annotations. Moreover, a new evaluation strategy is introduced to improve the reliability of the evaluation results. The details of the correlation evaluation and the new evaluation strategy are provided in Appendix A.18. For Math, we employ a rule-based extraction (Chen et al., 2023a) method to obtain accuracy. Utilizing the lm-evaluation-harness framework (Gao et al., 2021), we assess the accuracy of datasets in general domain knowledge (Gen-Kn.) and generic reasoning (Gen-Rs.). The instruction following (Instruct.) is evaluated using log-likelihood (LL) following the approach of Lin et al. (2023b). To evaluate overall performance in both speciality (Spec.) and versatility (Vers.), we define Spec. $= \frac{1}{3}(\text{BERTScore} + \text{BLEU} + \text{Rouge})$ in Finance, Law and CAG, while Spec. $=$ accuracy in Math. We define Vers. $= \frac{1}{3}(\text{Gen-Kn.} + \text{Gen-Rs.} + \text{Instruct.})$ and Uni. $=$ Spec. $+$ Vers., (Uni. $\in [0, 2]$). More details of evaluation metrics are available in Appendix A.10.

4.3 Results

Optimal Layer Range and Module (RQ1) The exploration algorithm mentioned in Sec. 3.3 is carried out on four tasks under 7B and 13B models. Due to space limits, we present overall results for the Finance task under the 13B model in Fig. 3 as an example, with detailed results available in Appendix A.12 for future analysis in this field.

As depicted in Fig. 3, during step 1, the optimal Uni. score is attained with the "(10, 30] - MHA & FFN" configuration. Notably, the Spec. scores in the bottom and middle layer ranges (e.g., (0, 20], (5, 25], and (10,30]) are relatively high and comparable, whereas the top layer range (e.g., (20, 40]) exhibits the lowest performance. Conversely, the Vers. score is lowest in the bottom layer range and improves as the layer range ascends. Similar findings can also be observed in the 7B model. Further analysis of this observation is provided in Sec. 4.4.

In step 2, the best Uni. score is achieved with "(10, 20] - MHA & FFN", where a reduction in the number of trainable layers from 20 to 10 results in an increase in Vers. score and a decrease in Spec. score. However, the decline in Spec. score

		7B				13B			
		Finance	Law	Math	CAG	Finance	Law	Math	CAG
-	*Full SFT*	0.8488	0.8460	0.5184	0.9315	0.8770	0.8901	0.5912	0.9999
Arch.	*LoRA*	0.8534	0.8744	0.5072	0.9656	0.8983	0.9226	0.5628	1.0215
Weight.	*Wise-FT*	0.8802	0.8898	0.5369	0.9745	0.9161	0.9313	0.6070	1.0310
	V-SoftMask	0.8537	0.8515	0.5186	0.9362	0.8807	0.8955	0.5954	1.0057
Regular.	*L1*	0.8580	0.8714	0.5122	0.9541	0.8988	0.9108	0.5733	1.0084
	L2	0.8475	0.8616	0.5135	0.9413	0.8772	0.8960	0.5808	1.0010
Ours	*CoFiTune*	**0.8901**	**0.8993**	**0.5597**	**0.9882**	**0.9296**	**0.9406**	**0.6250**	**1.0511**

Table 1: The Uni. scores of our *CoFiTune* and baseline methods in four tasks under the 7B and 13B models. Arch., Weight. and Regular. represent Architecture-, Weight-, and Regularization-based methods respectively.

	Finance		Math	
	Spec.	Vers.	Spec.	Vers.
ZeroShot	0.3766	0.5201	0.0760	0.5201
Full SFT	0.4761	0.4009	0.1400	0.4512
LoRA	0.4179	0.4804	0.0840	0.4788
Wise-FT	0.4308	0.4853	0.1000	0.5070
V-SoftMask	0.4752	0.4055	0.1400	0.4554
L1	0.4287	0.4701	0.0920	0.4813
L2	0.4368	0.4404	0.1080	0.4728
CoFiTune	0.4351	0.4945	0.1120	0.5130

Table 2: The Spec. and Vers. scores of Finance and Math tasks under the 13B model. *ZeroShot* denotes the aligned LLM without fine-tuning.

is less pronounced than the increase in Vers. score, summing up to a higher Uni. score in overall performance. Moving to step 3, the Uni. score for "(10, 20] - FFN" surpasses all other configurations. Specially, the Spec. score of "(10, 20] - FFN" is comparable to "(10, 20] - MHA & FFN" while its Vers. score is superior. Furthermore, the Spec. score of FFN significantly outperforms other modules including MHA, down_proj, and up_proj. The module importance for speciality will be further examined in Section 4.4.

Back to the whole picture, upon examining the results across various tasks and model scales, we surprisingly observe a consistent pattern:

Insight 1. *The "*$(\mathbf{N} \times \mathbf{25\%}, \mathbf{N} \times \mathbf{50\%}]$ *- FFN" configuration yields the best Uni. score on all tasks for both 7B and 13B models.*

Namely, the optimal Uni. score of the coarse-grained level search for all tasks under the 7B model (*i.e.*, $N = 32$) is achieved with "(8, 16] - FFN", while for the 13B model (*i.e.*, $N = 40$), it is "(10, 20] - FFN". Moreover, for certain special scenarios, we offer an optional solution in Appendix A.13.

Performance of *CoFiTune* (RQ2) We summarize the Uni. scores of our *CoFiTune* and the competitive methods across different tasks and model scales in Table 1. For further elaboration, the Spec. and Vers. scores for the Finance and Math tasks under the 13B model are presented in Table 2.

	CoFiTune		*CoFiTune w/o Fine-SoftMask*	
	Spec.	Vers.	Spec.	Vers.
Finance	0.4351	0.4945	0.4352	0.4912
Law	0.4503	0.4903	0.4512	0.4855
Math	0.1120	0.5130	0.1080	0.5102
CAG	0.5406	0.5105	0.5397	0.5054

Table 3: The impact of Fine-SoftMask in *CoFiTune* on Spec. and Vers. scores under the 13B model. "w/o" means excluding this technique from *CoFiTune*.

In Table 1, *CoFiTune* consistently outperforms all baseline methods. Specifically, in the Finance task of 7B model, it exhibits improvements in Uni. scores of 3.7%, 4.3%, and 4.5% compared to L1, LoRA, and V-SoftMask respectively. Table 2 highlights that *CoFiTune* especially succeeds in the balance of speciality and versatility (e.g., in Finance task, it reaches up to 98.1% of ZeroShot in Vers. score and up to 91.4% of Full SFT in Spec. score). In particular, Full SFT, V-SoftMask, and L2 exhibit a low Vers. score despite their relatively strong grasp of speciality. On the other hand, LoRA, L1, and Wise-FT fall short in terms of their performance in Spec. score. Similar trends are observed in the remaining results, detailed in Appendix A.15.

Impact of Fine-SoftMask (RQ3) We conduct additional experiments to demonstrate the effectiveness of the Fine-SoftMask mechanism discussed in Sec. 3.4, and give the conclusion below:

Insight 2. *Fine-SoftMask mechanism effectively mitigates the CF in LLM's versatility without harming the speciality performance.*

Concretely, we report the results under 13B in Table 3. When applying Fine-SoftMask, we observe nearly identical Spec. scores in the Finance and

	Finance	Law	Math	CAG
Full SFT	0.9294	0.9392	0.6910	1.0527
LoRA	0.9368	0.9635	0.6427	1.0648
Wise-FT	0.9595	0.9811	0.6876	1.0776
V-SoftMask	0.9343	0.9453	0.6911	1.0579
L1	0.9301	0.9536	0.6640	1.0527
L2	0.9170	0.9388	0.6698	1.0556
CoFiTune	**0.9722**	**0.9910**	**0.6991**	**1.0904**

Table 4: The Uni. score of our *CoFiTune* and baseline methods under the 33B model.

	3B		7B	
	Finance	Math	Finance	Math
Full SFT	0.7437	0.3548	0.7926	0.4365
LoRA	0.7514	0.3462	0.8073	0.4237
Wise-FT	0.7782	0.3715	0.8351	0.4519
V-SoftMask	0.7463	0.3564	0.7973	0.4383
L1	0.7568	0.3542	0.8144	0.4331
L2	0.7505	0.3547	0.8017	0.4304
CoFiTune	**0.7876**	**0.3891**	**0.8463**	**0.4758**

Table 5: The Uni. scores of Finance and Math tasks under the 3B and 7B BLOOMZ models.

Law tasks compared to not applying it, and even a slight improvement in CAG and Math tasks. Moreover, Fine-SoftMask contributes to a Vers. score improvement of 0.7%, 1%, and 1.1% in Finance, Law and CAG tasks respectively. Similar conclusions in the 7B model are presented in Table 10.

Performance of *CoFiTune* across Models of Different Scales and Families (RQ4) To verify the generalization of *CoFiTune* on different model scales and families, we directly apply Insight 1, obtained in **RQ1**, without tree-search exploration to the Chinese-Alpaca-Pro 33B model and another popular aligned model[9], BLOOMZ (Muennighoff et al., 2023). The overall results of the 33B model are shown in Table 4, with detailed results in Appendix A.15.3. For the BLOOMZ model, we conduct experiments on 3B and 7B scales under Finance and Math tasks, the overall results are presented in Table 5, with detailed results in Appendix A.14. As indicated in Table 4, *CoFiTune* maintains its superior unified performance under the 33B model, exhibiting improvements in Uni. score by 1.2%, 1.3%, and 1.6% in Finance, CAG, and Math tasks compared to the best-performing baseline Wise-FT. Similarly, Table 5 demonstrates the strong competence of our *CoFiTune* under the BLOOM family model, showcasing improvements in Uni. score by 5.9% and 6.4% in the Finance task when compared to Full SFT baseline. *These results further validate the widespread effectiveness of Insight 1 and CoFiTune across models of different scales and families.*[10]

4.4 Further Analysis

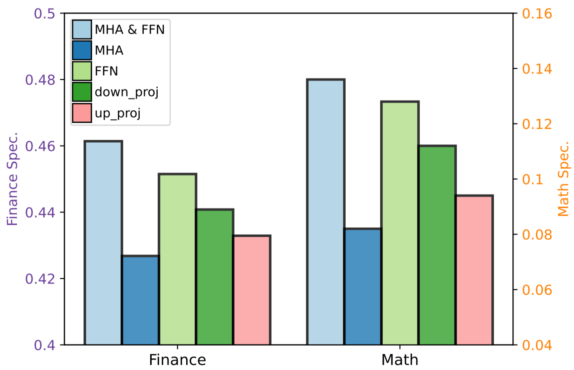

Figure 4: Spec. scores for Finance and Math tasks under the 13B model across modules trained in all layers.

Module Importance for Speciality To delve deeper into module significance for gaining speciality, we separately experimented with MHA, FFN, down_proj, and up_proj modules across all N layers in both the 7B and 13B models. Results for the Finance and Math tasks under the 13B model are shown in Fig. 4. As shown in Fig. 4, FFN achieves a Spec. score most comparable to MHA & FFN, followed by down_proj and up_proj, while MHA performs the least favorably. Similar trends are observed under the 7B model in Fig. 7. Based on these observations, we draw the following insight:

Insight 3. *FFN, especially the down_proj in it, is more crucial than MHA when gaining speciality.*

This[11] aligns, to some extent, with views from Geva et al. (2021) and Dai et al. (2022), suggesting FFN operates as key-value memories. The prediction distributions over the output vocabulary are determined by the down_proj, while MHA captures more superficial linguistic and pattern information (Manning et al., 2020; Rogers et al., 2021).

Exploring CF in LLM's Versatility We further explore the observation from Sec. 4.3 which unveils a notable gap in Vers. scores between the bottom $(0, N \times 50\%]$ and middle $(N \times 25\%, N \times 75\%]$

[9]In this paper, unless explicitly specified, the model referred to is Chinese-Alpaca-Pro.

[10]Due to the computing resource limitation, we defer the experimental analysis on even larger LLMs for future work.

[11]Notably, the number of trainable parameters in down_proj and MHA are of the same scale.

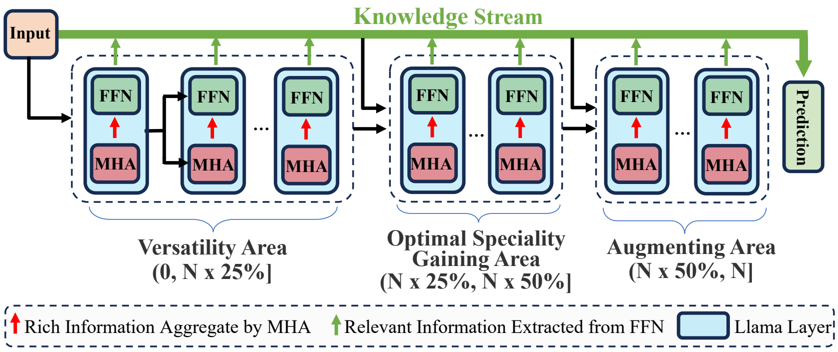

Figure 5: An illustration of our speculation on the process of information forwarding.

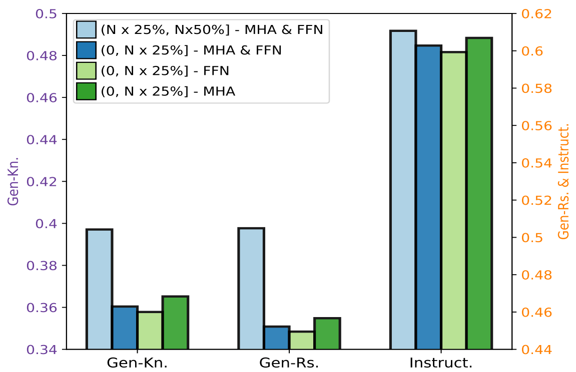

Figure 6: The Gen-Kn., Gen-Rs., and Instruct. scores in Math task under the 13B model.

layer ranges in both 7B and 13B models. Note that there are two non-overlapping layer ranges, $(0, N \times 25\%]$ and $(N \times 50\%, N \times 75\%]$, situated between $(0, N \times 50\%]$ and $(N \times 25\%, N \times 75\%]$. However, the Vers. score of $(N \times 50\%, N \times 75\%]$ is favorable (shown in Fig. 3 and Appendix A.12); therefore, we suspect that it is the $(0, N \times 25\%]$ that harms LLM's versatility.

Thus, we conduct experiments in this range with detailed Vers. score of Math task under 13B model in Fig. 6. In Fig. 6, we observe that: 1) The detailed Vers. scores of "$(0, N \times 25\%]$ - MHA & FFN" significantly lag behind "$(N \times 25\%, N \times 50\%]$ - MHA & FFN"; 2) In the layer range $(0, N \times 25\%]$, compared with "FFN & MHA", FFN further impairs versatility while MHA mitigates it. Similar findings across various tasks and model scales are shown in Appendix A.17. Hence, we derive the following insight:

Insight 4. *The LLM's versatility may predominantly reside in the layer range* $(0, N \times 25\%]$, *particularly within the FFN module.*

Furthermore, combining the hints from Geva et al. (2021), Meng et al. (2022b), Allen-Zhu and Li (2023b), Allen-Zhu and Li (2023a), and Chuang et al. (2023) with our findings in this work, we speculate the forward process in Fig. 5: 1) MHA modules in *Versatility Area* aggregate information, enriching input for corresponding FFN. FFN modules utilize this enriched input information to extract relevant knowledge, passing it through residual connections to *Optimal Speciality Gaining Area*; 2) *Optimal Speciality Gaining Area* handles both versatility and speciality information, forwarding it to *Augmenting Area*. 3) Processed information from the previous two areas is forwarded to *Augmenting Area* through residual connections, focusing on information enhancement for final prediction.

In this context, modifying modules in *Versatility Area* impairs the original memory and abilities of LLM, impeding the flow of versatility information passing through. In contrast, updating modules in *Optimal Speciality Gaining Area* allows versatility information to be recalled and passed through residual connections. Extra experiments are scheduled to validate these speculations in the future.

5 Conclusion

In this work, we strive to optimize the balance between LLM's speciality and versatility. Our proposed *CoFiTune* framework employs an empirical tree-search exploration algorithm to identify the module within a defined layer range that is crucial for gaining speciality without significantly affecting versatility. Additionally, the Fine-SoftMask mechanism is applied to further alleviate CF issue without impairing the speciality. We introduce a Chinese CF setting to advance the research in Chinese domain. The experimental results demonstrate that our *CoFiTune* outperforms all baselines across various tasks and model scales. Our in-depth analysis provides valuable insights for future work.

6 Limitations

In this study, we opt not to use the rehearsal-based method, which involves replaying a small portion of the general dataset during fine-tuning (Rolnick et al., 2019; de Masson D'Autume et al., 2019). This decision stems from the uncertainty surrounding the replay data ratio and strategy in this domain. Increasing the number of replay samples during fine-tuning would also incur higher training resource costs. Moreover, we believe that the rehearsal-based method can be integrated with our proposed *CoFiTune* framework, and we plan to explore this in future work.

Additionally, the model employed in this study is mainly based on the Llama architecture. This choice is driven by the fact that Llama family models demonstrate remarkable dominance and performance in the current NLP research area (constituting over 90% on the LLM leaderboard[12]). Besides, based on the fact that most current LLMs share the same architecture as Llama, i.e., transformer decoder-only architecture, we are confident that our identified universal optimal range can also be applied to a majority of models within the same architecture (the experiment results of BLOOMZ in **RQ4** have validated this point). Therefore, our *CoFiTune* framework can be directly adopted in these models using **Insight 1**, without incurring any tree-search exploration overhead. Furthermore, we plan to explore other model architectures (e.g., encoder-decoder) in future research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *CoRR, abs/2303.08774*.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023a. Physics of language models: Part 3.1, knowledge storage and extraction.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023b. Physics of language models: Part 3.2, knowledge manipulation.

Bang An, Jie Lyu, Zhenyi Wang, Chunyuan Li, Changwei Hu, Fei Tan, Ruiyi Zhang, Yifan Hu, and Changyou Chen. 2020. Repulsive attention: Rethinking multi-head attention as bayesian inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 236–255.

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, S. Bodapati, Katrin Kirchhoff, and Dan Roth. 2023. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. In *Annual Meeting of the Association for Computational Linguistics*.

[12]https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

FeiDa Chen. 2018. The legal consultation data and corpus of the thesis from China law network.Replication Data for: Design and research of legal consultation text classification system.

Nuo Chen, Zinan Zheng, Ning Wu, Linjun Shou, Ming Gong, Yangqiu Song, Dongmei Zhang, and Jia Li. 2023a. Breaking language barriers in multilingual mathematical reasoning: Insights and observations. *CoRR, abs/2310.20246*.

Wei Chen, Qiushi Wang, Zefei Long, Xianyin Zhang, Zhongtian Lu, Bingxuan Li, Siyuan Wang, Jiarong Xu, Xiang Bai, Xuanjing Huang, et al. 2023b. Disc-finllm: A chinese financial large language model based on multiple experts fine-tuning. *CoRR, abs/2310.15205*.

Daixuan Cheng, Shaohan Huang, and Furu Wei. 2023. Adapting large language models via reading comprehension. *CoRR, abs/2309.09530*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *CoRR, abs/2309.03883*.

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*.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023a. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *CoRR, abs/2306.16092*.

Yiming Cui, Ziqing Yang, and Xin Yao. 2023b. Efficient and effective text encoding for chinese llama and alpaca. *CoRR, abs/2304.08177*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32.

Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023. Lmflow: An extensible toolkit for finetuning and inference of large foundation models. *CoRR, abs/2306.12420*.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *CoRR, abs/2310.05492*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

Denis Emelin and Rico Sennrich. 2021. Wino-X: Multilingual Winograd schemas for commonsense reasoning and coreference resolution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8517–8532.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR, abs/2312.10997*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *CoRR, abs/2309.17452*.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. DuReader: a Chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 37–46. Association for Computational Linguistics.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kuebler, and Larry Moss. 2020. Ocnli: Original chinese natural language inference. In *Findings of EMNLP*.

Quzhe Huang, Mingxu Tao, Zhenwei An, Chen Zhang, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. 2023. Lawyer llama technical report. *CoRR, abs/2305.15062*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Shibo Jie, Zhi-Hong Deng, and Ziheng Li. 2022. Alleviating representational shift for continual fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3810–3819.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2022. Continual pre-training of language models. In *The Eleventh International Conference on Learning Representations*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR, abs/1412.6980*.

Cunliang Kong, Yun Chen, Hengyuan Zhang, Liner Yang, and Erhong Yang. 2022a. Multitasking framework for unsupervised simple definition generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5934–5943.

Cunliang Kong, Yujie Wang, Ruining Chong, Liner Yang, Hengyuan Zhang, Erhong Yang, and Yaping Huang. 2022b. Blcu-icall at semeval-2022 task 1: Cross-attention multitasking framework for definition modeling. *arXiv preprint arXiv:2204.07701*.

Dawei Li, Hengyuan Zhang, Yanran Li, and Shiping Yang. 2023a. Multi-level contrastive learning for script-based character understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5995–6013.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023b. Cmmlu: Measuring massive multitask language understanding in chinese.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. In *First Workshop on Interpolation Regularizers and Beyond at NeurIPS 2022*.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023c. Large language models in finance: A survey. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 374–382.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2023a. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*.

Yong Lin, Lu Tan, Hangyu Lin, Zeming Zheng, Renjie Pi, Jipeng Zhang, Shizhe Diao, Haoxiang Wang, Han Zhao, Yuan Yao, et al. 2023b. Speciality vs generality: An empirical study on catastrophic forgetting in fine-tuning foundation models. *CoRR, abs/2309.06256*.

Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2947–2962.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021a. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Tian Yu Liu and Stefano Soatto. 2023. Tangent model composition for ensembling and continual fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18676–18686.

Zihan Liu, Genta Indra Winata, and Pascale Fung. 2021b. Continual mixed-language pre-training for extremely low-resource neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2706–2718.

Jinghui Lu, Rui Zhao, Brian Mac Namee, and Fei Tan. 2023a. Punifiedner: A prompting-based unified ner system for diverse datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13327–13335.

Jinghui Lu, Dongsheng Zhu, Weidong Han, Rui Zhao, Brian Mac Namee, and Fei Tan. 2023b. What makes pre-trained language models better zero-shot learners? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2288–2303.

Yun Luo, Zhen Yang, Xuefeng Bai, Fandong Meng, Jie Zhou, and Yue Zhang. 2023a. Investigating forgetting in pre-trained representations through continual learning. *CoRR, abs/2305.05968*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023b. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *CoRR, abs/2308.08747*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *EMNLP 2021-2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. 2024. ReLU strikes back: Exploiting activation sparsity in large language models. In *The Twelfth International Conference on Learning Representations*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, et al. 2023. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15991–16111.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2022. Crosslingual generalization through multitask finetuning.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. 2023. Proving test set contamination in black box language models. *CoRR, abs/2310.17623*.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *CoRR, abs/2312.05934*.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. *CoRR, abs/2302.06600*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *CoRR, abs/2304.03277*.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

Rylan Schaeffer. 2023. Pretraining on the test set is all you need. *CoRR, abs/2309.08632*.

Noam Shazeer. 2020. Glu variants improve transformer. *CoRR, abs/2002.05202*.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. 2023. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *CoRR, abs/2304.06027*.

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

Fei Tan, Yifan Hu, Changwei Hu, Keqian Li, and Kevin Yen. 2020. TNT: Text normalization based pre-training of transformers for content moderation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4735–4741, Online. Association for Computational Linguistics.

Fei Tan, Yifan Hu, Kevin Yen, and Changwei Hu. 2021. BERT-beta: A proactive probabilistic approach to text moderation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8667–8675, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. `https://github.com/tatsu-lab/stanford_alpaca`.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning.

Yongqi Tong, Yifan Wang, Dawei Li, Sizhe Wang, Zi Lin, Simeng Han, and Jingbo Shang. 2023. Eliminating reasoning via inferring with planning: A new framework to guide llms' non-linear thinking. *arXiv preprint arXiv:2310.12342*.

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. *CoRR, abs/2302.13971*.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, et al. 2023a. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *CoRR, abs/2310.07521*.

Neng Wang, Hongyang Yang, and Christina Wang. 2023b. Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. 2023c. Trace: A comprehensive benchmark for continual learning in large language models. *CoRR, abs/2310.06762*.

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

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, et al. 2023d. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10933–10946.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali

Farhadi, Hongseok Namkoong, et al. 2022. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971.

Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. 2021. Robust fine-tuning of zero-shot models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7949–7961.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. 2021. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528.

LI Xuhong, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pages 2825–2834. PMLR.

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

Hengyuan Zhang, Dawei Li, Yanran Li, Chenming Shang, Chufan Shi, and Yong Jiang. 2023. Assisting language learners: Automated trans-lingual definition generation via contrastive prompt learning. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 260–274.

Hengyuan Zhang, Dawei Li, Shiping Yang, and Yanran Li. 2022a. Fine-grained contrastive learning for definition generation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1001–1012.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022b. Continual sequence generation with adaptive compositional modules. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3653–3667.

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. *CoRR, abs/2303.18223*.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. Jecqa: A legal-domain question answering dataset. In *Proceedings of AAAI*.

A Appendix

A.1 Detailed Information of Llama Backbone

The MHA operation is a crucial component of the transformer, defined as:

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O \tag{7}$$

where ; is the concatenation operation, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, and $\mathbf{W}^O$ is a learnable output matrix unique to the MHA module. Each head is computed as:

$$\begin{aligned} \text{head}_i &= \text{Attention}(\mathbf{x}\mathbf{W}_i^Q, \mathbf{x}\mathbf{W}_i^K, \mathbf{x}\mathbf{W}_i^V) \\ \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) &= \text{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i \end{aligned} \tag{8}$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$ are the corresponding weight matrices of query, key and value for the i-th head.

The FFN module is parameterized by an up projection matrix (up_proj), followed by a down projection matrix (down_proj):

$$\begin{aligned} &\text{FFN}(\mathbf{x}') = \big(\text{SwiGLU}(\mathbf{x}') \otimes (\mathbf{x}'\mathbf{W}^1)\big)\mathbf{W}^2 \\ &\text{SwiGLU}(\mathbf{x}') = \text{SiLU}(\mathbf{x}'\mathbf{W}) \\ &\text{RMSNorm}(\mathbf{x}') = \frac{w \odot \mathbf{x}'}{\sqrt{\text{Var}(\mathbf{x}') + \epsilon}} \end{aligned} \tag{9}$$

where $\mathbf{x}'$ is the input tensor of FFN, $\mathbf{W}$ is the learnable weight matrix without bias (a.k.a gated projection matrix), $\mathbf{W}^1$ and $\mathbf{W}^2$ are up_proj and down_project respectively. $\text{SiLU}(\mathbf{x}') = \mathbf{x}' \otimes \text{Sigmoid}(\mathbf{x}')$. $\otimes$ is the element-wise multiplication operation. w represents the learnable weight parameter, $\text{Var}(\mathbf{x}')$ denotes the variance of $\mathbf{x}'$ across the last dimension, and ϵ is a small constant introduced for numerical stability.

As the Llama layer lacks dropout module, we manually add it after the output of down_proj and the attention weights of MHA when computing the importance of units in Sec. 3.4.

Given Mirzadeh et al. (2024)'s demonstration that SwiGLU, a commonly used trainable activation function in the FFN of LLMs, can be substituted with non-trainable ReLU to decrease computation without compromising performance, we have chosen not to investigate SwiGLU in our exploration outlined in Sec. 3.3. Specifically, our exploration focuses solely on the up_proj and down_proj components within the FFN.

A.2 Smaller Exploring Layer Range Results

	7B		13B	
	Finance	Math	Finance	Math
(0, N] MHA & FFN	0.4432	0.0960	0.4614	0.136
(N x 25%, N x 37.5%] - MHA & FFN	0.3921	0.0360	0.4183	0.068
(N x 37.5%, N x 50%] - MHA & FFN	0.3923	0.0400	0.4189	0.064

Table 6: The speciality performance of Finance and Math under 7B and 13B within a smaller layer range.

There is a substantial speciality gap between the smaller layer range and the all-layer range. For example, in Math under the 7B model, the smaller layer range $(N \times 25\%, N \times 37.5\%$ accounts for only 37.5% of $(0, N]$ in Accuracy.

A.3 Exploration Algorithm in Coarse-grained Level

The exploration at coarse level is illustrated in Algorithm 1. The variables $results_1, results_2, results_3$ are dictionary data structures intended to store results from steps 1, 2, and 3, respectively. The variables $left, right, pivot$ represent the start index of their corresponding layer range. α denotes the number of layers targeted for coverage. For instance, in a 13B model ($N = 40$), $\alpha = 20$ in step 1 and $\alpha = 10$ in step 2. The GetBestLayers function aims to identify the layer range that performs best in terms of the unified score. The GetSubModules function returns the sub-modules of a specific module; for example, the sub-modules of FFN are up_proj and down_proj. The functions GetMax and GetSecond are responsible for obtaining the start index of the layer range that achieves the best and second-best performance. $f_{*\_layer\_range}$ denotes the parameters of FFN and MHA in the $*\_layer\_range$ that are trainable in the model, while others are frozen. Similarly, $f^{m}_{*\_layer\_range}$ represents the parameters of a specific module in the $*\_layer\_range$ that are trainable in the model, while others are frozen. The Max function returns the optimal strategy that achieves the highest unified score among all three steps.

Algorithm 1 Exploration Algorithm in Coarse Level

Input: Llama Model $f(\cdot)$, N layers, evaluation data D
Initialize: $results_1, results_2, results_3 \leftarrow \{\}$, $left \leftarrow 0$, $right \leftarrow N \times 50\%, \alpha \leftarrow N \times 50\%$
Function:
def GetBestLayerRange($left$, $right$, α, $result$):
    $pivot \leftarrow (left + right)//2$
    $left\_layer\_range \leftarrow (left, left + \alpha]$
    $right\_layer\_range \leftarrow (right, right + \alpha]$
    $pivot\_layer\_range \leftarrow (pivot, pivot + \alpha]$
    $result[left] \leftarrow$ Eval($f_{left\_layer\_range}(\cdot), D$)
    $result[right] \leftarrow$ Eval($f_{right\_layer\_range}(\cdot), D$)
    $result[pivot] \leftarrow$ Eval($f_{pivot\_layer\_range}(\cdot), D$)
    $left \leftarrow$ GetMax($result$)
    $right \leftarrow$ GetSecond($result$)
    $pivot_2 \leftarrow (left + right)//2$
    $pivot_2\_layer\_range \leftarrow (pivot_2, pivot_2 + \alpha]$
    $result[pivot_2] \leftarrow$ Eval($f_{pivot_2\_layer\_range}(\cdot), D$)
    return GetMax($result$)
 # Step 1:
$lr\_broad\_start\_idx \leftarrow$ GetBestLayerRange($left$, $right$, α, $results_1$)
 # Step 2:
$left \leftarrow lr\_broad\_start\_idx$; $right \leftarrow lr\_broad\_start\_idx + \alpha$; $\alpha \leftarrow \alpha//2$
$lr\_narrow\_start\_idx \leftarrow$ GetBestLayerRange($left$, $right$, α, $results_2$)
 # Step 3:
$lr\_narrow \leftarrow (lr\_narrow\_start\_idx, lr\_narrow\_start\_idx + \alpha]$
for $m \leftarrow$ [MHA, FFN] **do**
    $results_3[m] \leftarrow$ Eval($f^{m}_{lr\_narrow}(\cdot), D$)
end for
$best\_module \leftarrow$ Max($results_3$)
for $m_{sub} \leftarrow$ GetSubModules($best\_module$) **do**
    $results_3[m_{sub}] \leftarrow$ Eval($f^{m_{sub}}_{lr\_narrow}(\cdot), D$)
end for
return Max($results_1, results_2, results_3$)

A.4 Discussion of our setting

Recall the conventional settings in continual learning, where a pre-trained model f_{θ_0} undergoes sequential fine-tuning on tasks [$\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_K$], and forgetting evaluation is based on the model's performance on $\mathcal{T}_i$ after being trained on $\mathcal{T}_j$ $(i < j)$.

In contrast, our investigation focuses solely on forgetting in the versatility of the original model during the fine-tuning of a single task. This setting is motivated by two key considerations:

- Aligned LLMs have demonstrated remarkable versatility, achieved through extensive pre-training on a massive dataset comprising tens of billions to even hundreds of billions of samples. This enables them to effectively handle a wide range of real-world tasks (Chowdhery et al., 2023; Achiam et al., 2023). However, adopting a sequential fine-tuning method poses a significant risk of progressively undermining the model's versatility at each fine-tuning step (Luo et al., 2023b; Wang et al., 2023c). Additionally, the severe decline in versatility not only inhibits the fine-tuned performance of the model on diverse tasks (Cheng et al., 2023; Dong et al., 2023) but also hinders the effectiveness of transferring knowledge from it when gaining speciality (Jie et al., 2022; Luo et al., 2023a).

- Recent studies suggest an alternative to the traditional sequential fine-tuning approach. Instead of fine-tuning tasks sequentially, each task is independently fine-tuned on a pre-trained model, and their weights are subsequently combined through interpolation (Ilharco et al., 2022; Li et al., 2022; Liu and Soatto, 2023; Yu et al., 2023). This weight interpolation approach significantly enhances performance and mitigates catastrophic forgetting compared to conventional methods that sequentially fine-tune one model on $[\mathcal{T}^1, \mathcal{T}^2, \mathcal{T}^3 ...]$.

A.5 Baseline Descriptions

In this Section, we describe the baseline method in our setting in detail. We carefully categorize them into three classes:

A.5.1 Regularization-based Methods

L1 Regularization Panigrahi et al. (2023) introduced explicit L1 regularization (with a strength of 0.001) on the parameter shift $\theta \to \hat{\theta}$, denoted as $|\hat{\theta} - \theta|$. This regularization strategy helps alleviate catastrophic forgetting by limiting the extent of tuning.

L2 Regularization Building upon the concept introduced by Xuhong et al. (2018), Lin et al. (2023b) utilized a more consistent parameter regularization approach. Specifically, the divergence between the parameters of the fine-tuned model $\hat{\theta}$ and the pre-trained model θ serves as the object of the L2 penalty in the optimization process, denoted as $|\hat{\theta} - \theta|_2^2$. This approach helps prevent deviation from the pre-trained model during tuning.

A.5.2 Weight-based Methods

Wise-FT The Wise-FT (Wortsman et al., 2021) methodology involves two distinct stages: initially fine-tuning the pre-trained model θ on a specific downstream task to obtain the fine-tuned model $\hat{\theta}$, followed by the fusion of the original pre-trained model θ and the fine-tuned model $\hat{\theta}$ using linear weight interpolation $f_{(1-\alpha)\theta+\alpha\hat{\theta}}$, where α is a hyper-parameter ranging from 0 to 1. Parameter Constraint-Based Methods formulate their approach around parameter-specific strategies, leveraging either parameter efficiency techniques or the estimation of parameter importance.

Vanilla Soft-masking Ke et al. (2022) proposed Vanilla Soft-masking to address CF in continual domain pre-training of language models. Specifically, a gradient-based detection method is used to compute the importance value of units within the attention and FFN modules across all transformer layers for general domain knowledge. The resulting importance vector is then employed to control the backward gradient flow. Importantly, the soft-masking regulation is exclusively applied to the backward process, ensuring that knowledge transfer across domains during tuning remains unaffected.

A.5.3 Architecture-based Methods

LoRA Grounded in the assumption that the alteration in weights during model adaptation exhibits a low "intrinsic rank," Hu et al. (2021) introduced a low-rank matrix into the dense layers within the network. This allows the indirect training of these layers by optimizing the decomposition matrices within the low-rank. Throughout training, all the decomposition matrices within the low-rank matrix are trainable, while the pre-trained weights remain frozen to preserve the original general ability of the pre-trained model.

A.6 Statistics and Descriptions of Datasets

	Dataset Name	Train	Test	Avg. Length
Knowledge Intensive Task	Finance	14337	600	270.94
	Law	19400	600	277.82
Ability Intensive Task	Math	7366	250	216.95
	CAG	19400	600	331.24
General Domain Knowledge	CMMLU	-	11582	71.89
Generic Reasoning	LogiQA	-	645	238.91
	LogiQA2	-	1589	224.25
	OCNLI	-	1995	82.17
	Zh-Winograd	-	499	65.59
Instruction Following	Chinese GPT4-Instruct	-	1000	210.18

Table 7: The statistical information of the datasets involved in our setting. "Train" denotes the number of samples in the training set, "Test" denotes the number of samples in the test set, and "Avg. Length" denotes the average length of the dataset.

Currently, LLM still falls short in certain areas (Li et al., 2023a; Lu et al., 2023a), we mainly select two types of tasks to evaluate its speciality. The overall statistics of the selected task datasets are shown in Table 7, and the detailed descriptions are as follows:

A.6.1 Knowledge Intensive Tasks

- **Finance:** To create a Chinese financial QA instruction format dataset, we adopt the building procedure from Chen et al. (2023b). Initially, we translate the instruction content of the English FiQA[13] dataset into Chinese. Subsequently, we employ the translated Chinese instructions to generate responses that align with China's national conditions using a robust Chinese FinLM[14] (Chen et al., 2023b).

- **Law:** Huang et al. (2023) initially gather Chinese legal questions from OpenLaw (Chen, 2018) and JEC-QA (Zhong et al., 2020). They then employ ChatGPT to obtain responses, constructing the LlamaLawyer[15] dataset. This dataset comprises both single-turn and multi-turn instructions. For a fair comparison, we exclusively choose single-turn instructions in our setting.

A.6.2 Ability Intensive Tasks

- **Math:** Chen et al. (2023a) establishes a multi-lingual GSM8k dataset (MGSM8k)[16] by translating the original English GSM8k (Cobbe et al., 2021) dataset into ten different languages using ChatGPT. In our setting, we specifically select the Chinese version within MGSM8k for both training and evaluation.

[13] https://huggingface.co/datasets/FinGPT/fingpt-fiqa_qa
[14] https://huggingface.co/Go4miii/DISC-FinLLM
[15] https://github.com/AndrewZhe/lawyer-llama
[16] https://mathoctopus.github.io/

- **Context-Aware Generation (CAG[17]):** To enhance the model's capability of leveraging the context information to generate answers (Tong et al., 2023, 2024), we follow the methodology RA-DIT mentioned in Lin et al. (2023a) to construct an instruction format dataset based on Dureader (He et al., 2018), a Chinese Reading Comprehension (RC) dataset that includes three types of questions: "Entity", "Description" and "YesNo". In our setting, we specifically choose "Description" type questions (Dureader-Desc), because their answers are human-written, presenting a significant difference from the context content. Following the methodology of Chen et al. (2023b), we design a prompt (illustrated in Appendix A.9.2) that allows the model to flexibly generate appropriate responses under contexts with different properties. We select the context that is irrelevant to the corresponding question to form a negative sample. This aims to enable the LLM to ignore misleading context content and lean into its parametric knowledge to respond. In contrast, we form positive samples by selecting context relevant to the corresponding question, intending to evaluate whether the LLM can better utilize relevant background knowledge to make a prediction. During the evaluation of the Context-Aware Generation (CAG) task, we use a subset of negative samples from the training data, paraphrase their instructions during inference, and utilize automatic generation metrics to assess the model's ability to leverage its parametric knowledge for response. We use positive samples outside the training data to evaluate the model's proficiency in utilizing context for response.

A.6.3 General Domain knowledge

We employ CMMLU (Li et al., 2023b) for a zero-shot evaluation of the model's general domain knowledge. CMMLU[18] covers a wide range of subjects, comprising 67 topics that span from elementary to advanced professional levels. It includes subjects that require computational expertise, such as physics and chemistry, as well as disciplines within humanities and social sciences. Notably, many tasks within CMMLU involve contextual nuances and wording that may not easily translate across languages. Moreover, numerous tasks have answers specific to China, making them contextually relevant but potentially not universally applicable or considered correct in other regions or languages.

A.6.4 Generic Reasoning

We assess the generic reasoning ability of LLM through the following datasets in a zero-shot manner:

- **LogiQA[19]:** The dataset follows a paragraph-question pair format, each accompanied by four candidate answers. It is sourced from publicly available logical examination papers for reading comprehension, designed by domain experts to assess participants' logical reasoning ability. Therefore, the questions exhibit reliable quality and cover a diverse range of topics.

- **LogiQA2[20]:** This is the second version of the LogiQA dataset, collected from the Chinese Civil Service Entrance Examination. The dataset includes newly released exam questions and practice questions, sourced from approximately 20 provinces in China where the exam is held annually. Exam materials are made publicly available on the Internet after each exam, and practice questions are obtained from various sources.

- **OCNLI[21]:** The Original Chinese Natural Language Inference dataset (OCNLI) is the first extensive Natural Language Inference (NLI) dataset for Chinese. Unlike other datasets, OCNLI does not depend on automatic translation or non-expert annotation. Instead, it gathers annotations from native speakers with expertise in linguistics.

- **Zh-Winograd:** XWinograd (Emelin and Sennrich, 2021)[22] is a multilingual Winograd dataset

[17] The CAG task essentially serves as the generation step in Retrieval-Augmented Generation (RAG), flexibly relying on the retrieved context to generate responses.

[18] https://huggingface.co/datasets/haonan-li/cmmlu

[19] https://github.com/lgw863/LogiQA-dataset

[20] https://github.com/csitfun/LogiQA2.0

[21] https://github.com/CLUEbenchmark/OCNLI/

[22] https://huggingface.co/datasets/Muennighoff/xwinograd

designed for assessing commonsense reasoning and coreference resolution. In our setting, we exclusively choose the Chinese version for evaluation.

A.6.5 Instruction Following Dataset

Chinese GPT4-Instruct encompasses a variety of user-oriented instructions, spanning areas such as email writing, social media, and entertainment. This dataset was introduced by Peng et al. (2023), who rigorously followed the methodology outlined by Taori et al. (2023) to implement the self-instruct strategy (Wang et al., 2022) with GPT-4.

A.7 Implementation Details

	7B			**13B**			**33B**		
	Epochs	Batch	LR	Epochs	Batch	LR	Epochs	Batch	LR
Finance	3	256	4e-5	5	256	4e-5	5	256	2e-5
Law	3	256	4e-5	5	256	4e-5	5	256	2e-5
Math	3	128	2e-5	5	128	2e-5	5	128	1e-5
RAG	3	256	4e-5	5	256	4e-5	5	256	2e-5

Table 8: The Hyper-parameters involved in our training setting. LR denotes "learning rate", Batch denotes "global batch size".

We use *DeepSpeed Zero3* (Rajbhandari et al., 2021) and Adam optimizer (Kingma and Ba, 2014) to train all the models. Our experiments are conducted on a computing platform equipped with 32 V100 GPUs. Each experiment is run with three different random seeds, and the results are averaged to obtain the final outcome. The maximum sequence length is set to 512 across all four tasks, and we employ a greedy decoding strategy for generating results. A cosine scheduler with a 3% warm-up period is applied. In our study, LoRA is applied to five weight matrices: $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ in the MHA module, and $\mathbf{W}_{down}$, $\mathbf{W}_{up}$ in the FFN module. Following Lin et al. (2023b), we experiment with $lora\_rank$ values in [4, 8, 16], setting $lora\_alpha = 2 \times lora\_rank$ to achieve optimal performance. For Wise-FT, consistent with Lin et al. (2023b), we explore α values in [0.4, 0.6, 0.8] to determine the best performance. Different learning rates and global batch sizes are set for training tasks across various model sizes, with detailed information provided in Table 8.

A.8 Details of Different Model Scales

All the model adheres to the Apache-2.0 license, the details of different model scales are illustrated in Table 9.

	Dimension	Heads	Layers
7B	4096	32	32
13B	5120	40	40
33B	6656	52	60

Table 9: The detailed architectural information for Chinese-Alpaca-Pro across various model scales.

A.9 Prompts

A.9.1 Instruction Prompt

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
{instruction}
{input}

Response: {output}

We use "{instruction}", "{input}", and "{output}" to replace the specific instruction, input, and output.

A.9.2 CAG Prompt

请根据参考材料回答下面的问题。下面的材料可供参考。
"Please answer the following questions based on the reference materials. The following materials are available for reference."
（注意：*1*、材料可能与问题无关，请忽略无关材料，并基于已有知识回答问题。*2*、尽量不要直接复制材料内容作为答案，而应该将材料内容作为事件的补充与潜在分析，启发思考）；
"(Note: 1. The material may have nothing to do with the question. Please ignore the irrelevant material and answer the question based on existing knowledge. 2. Try not to directly copy the material content as an answer, but use the material content as a supplement and potential analysis of the event to inspire thinking.);"
参考材料*:*
"references:"
{reference}
问题*:*
"question:"
{question}

Here, the contents labeled with "" contain the corresponding English translations of Chinese. We use "{reference}" and "{question}" to replace the specific reference and question.

A.10 Evaluation Metrics

A.10.1 Instruction Following Score

Following LMflow (Diao et al., 2023), we assess instruction following performance using log-likelihood:

$$\begin{aligned}\text{NLL} &= -\frac{1}{N}\sum_{i=1}^{N}\log p(\text{sentence}_i|\text{context}_i)\\ &= -\frac{1}{N}\sum_{i=1}^{N}\log p(\text{token}_{i,1},\text{token}_{i,2},...,\text{token}_{i,n_i}|\text{context}_i),\end{aligned}$$

where n_i is the length of the token in sentence$_i$. As the LL results are typically in the order of several hundreds (e.g., 531.27), and the results of Accuracy or automatic generation metrics are values ranging from 0 to 1, they inherently possess disparate scales. To comprehensively assess both the speciality and versatility of LLM, we initially scale the LL by dividing it by one thousand. Since a lower LL is indicative of superior performance, we then subtract the result from 1 to obtain the final instruction following (Instruct.) score, i.e., Instruct. $= 1 - \text{LL}/1000$.

A.10.2 Speciality Score

For Finance, Law, and CAG tasks, we follow Tan et al. (2020, 2021); Zhang et al. (2022a); Kong et al. (2022b,a); Zhang et al. (2023) to utilize the automatic metrics to evaluate the quality of generated text , i.e., BERTScore[23], BLEU[24], and Rouge[25] to evaluate their speciality performance. The Rouge metric comprises Rouge-1, Rouge-2, and Rouge-L. We initially average the results of Rouge-1, Rouge-2, and Rouge-L to obtain the overall Rouge result. Subsequently, we further average the results of BERTScore, BLEU, and Rouge to derive the final speciality score. It's worth noting that we employ the F1 score in both Rouge and BERTScore.

$$\begin{aligned} \text{Rouge} &= \frac{\text{Rouge-1} + \text{Rouge-2} + \text{Rouge-L}}{3} \\ \text{Spec.} &= \frac{\text{BERTScore} + \text{Rouge} + \text{BLEU}}{3} \end{aligned} \tag{10}$$

For Math, as it only has one metric, i.e., Accuracy, we directly use its Accuracy result as the Spec. score.

A.10.3 Versatility Score

We assess the versatility of LLM in three aspects: general domain knowledge (Gen-Kn.), generic reasoning (Gen-Rs.), and instruction following (Instruct.). For evaluating the generic reasoning of LLM, we average the results from four datasets to obtain an overall generic reasoning (Gen-Rs.) score. Subsequently, we average the scores of Gen-Kn., Gen-Rs., and Instruct. (computed in Appendix A.10.1) to derive the overall versatility (Vers.) score for LLM.

$$\begin{aligned} \text{Gen-Rs.} &= \frac{\text{LogiQA} + \text{LogiQA2} + \text{OCNLI} + \text{Zh-Winograd}}{4} \\ \text{Vers.} &= \frac{\text{Gen-Kn.} + \text{Gen-Rs.} + \text{Instruct.}}{3} \end{aligned} \tag{11}$$

A.10.4 Unified Score

To facilitate the evaluation of LLM in both speciality and versatility, we sum the results of the speciality (Spec.) score (obtained in Appendix A.10.2) and the versatility (Vers.) score (obtained in Appendix A.10.3) to get the final unified (Uni.) score:

$$\text{Uni.} = \text{Spec.} + \text{Vers.} \tag{12}$$

[23] https://github.com/Tiiiger/bert_score
[24] https://www.nltk.org/api/nltk.translate.bleu_score.html
[25] https://github.com/pltrdy/rouge

A.11 Ablation Results of Fine-SoftMask

	CoFiTune		*CoFiTune w/o Fine-SoftMask*	
	Spec.	Vers.	Spec.	Vers.
Finance	0.4218	0.4683	0.4220	0.4653
Law	0.4328	0.4665	0.4331	0.4626
Math	0.0800	0.4797	0.0800	0.4765
CAG	0.5166	0.4716	0.5159	0.4684

Table 10: The impact of Fine-SoftMask in *CoFiTune* on Spec. and Vers. scores under 7B model. “w/o” means excluding the technique from *CoFiTune*.

A.12 Exploration Results in Coarse Level

A.12.1 7B Results

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
(0, 16] - MHA & FFN	0.7283	0.3774	0.1843	0.5314	0.2512	0.3498	0.4300
(8, 24] - MHA & FFN	0.7298	0.3814	0.1884	0.5339	0.2575	0.3528	0.4332
(16, 32] - MHA & FFN	0.7162	0.3558	0.1649	0.5048	0.2324	0.3303	0.4123
(4, 20] - MHA & FFN	0.7306	0.3799	0.1887	0.5314	0.2562	0.3521	0.4331
(8, 16] - MHA & FFN	0.7266	0.3721	0.1807	0.5262	0.2466	0.3434	0.4264
(16, 24] - MHA & FFN	0.7151	0.3524	0.1629	0.5048	0.2273	0.3251	0.4101
(12, 20] - MHA & FFN	0.7217	0.3633	0.1714	0.5179	0.2398	0.3322	0.4188
(10, 18] - MHA & FFN	0.7228	0.3671	0.1742	0.5214	0.2425	0.3378	0.4213
(8, 16] - FFN	0.7219	0.3643	0.1797	0.5196	0.2377	0.3357	0.4220
(8, 16] - MHA	0.7151	0.3488	0.1603	0.5028	0.2221	0.3214	0.4081
(8, 16] - down_proj	0.7172	0.3569	0.1686	0.5095	0.2296	0.3287	0.4143
(8, 16] - up_proj	0.7159	0.3507	0.1643	0.5061	0.2253	0.3209	0.4103

Table 11: The detailed speciality performance of Finance task under the 7B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 16] - MHA & FFN	0.5375	0.3172	0.4269	0.2496	0.2801	0.4547	0.7234	0.4272
(8, 24] - MHA & FFN	0.5447	0.3524	0.4409	0.2775	0.3255	0.4503	0.7104	0.4460
(16, 32] - MHA & FFN	0.5579	0.3609	0.4535	0.2759	0.3386	0.4461	0.7535	0.4574
(4, 20] - MHA & FFN	0.5426	0.3414	0.4338	0.2682	0.3033	0.4484	0.7152	0.4392
(8, 16] - MHA & FFN	0.5770	0.3518	0.4437	0.2774	0.3158	0.4464	0.7356	0.4575
(16, 24] - MHA & FFN	0.5858	0.3609	0.4515	0.2729	0.3361	0.4536	0.7434	0.4661
(12, 20] - MHA & FFN	0.5792	0.3545	0.4461	0.2813	0.3193	0.4406	0.7432	0.4599
(10, 18] - MHA & FFN	0.5779	0.3531	0.4442	0.2801	0.3186	0.4390	0.7389	0.4583
(8, 16] - FFN	0.5999	0.3533	0.4428	0.2791	0.3055	0.4415	0.7451	0.4653
(8, 16] - MHA	0.6128	0.3502	0.4468	0.2862	0.3103	0.4422	0.7485	0.4699
(8, 16] - down_proj	0.6085	0.3537	0.4453	0.2915	0.2995	0.445	0.7457	0.4692
(8, 16] - up_proj	0.6147	0.3566	0.4475	0.3023	0.2991	0.4455	0.7433	0.4729

Table 12: The detailed versatility performance of Finance task under the 7B model in different exploring layer ranges and modules.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
(0, 16] - MHA & FFN	0.7313	0.3846	0.2092	0.5198	0.2579	0.3762	0.4417
(8, 24] - MHA & FFN	0.7341	0.3903	0.2152	0.5321	0.2645	0.3843	0.4465
(16, 32] - MHA & FFN	0.7200	0.3621	0.1739	0.4951	0.2363	0.3550	0.4187
(4, 20] - MHA & FFN	0.7329	0.3874	0.2126	0.5179	0.2615	0.3828	0.4443
(8, 16] - MHA & FFN	0.7293	0.3848	0.2007	0.5184	0.2590	0.3772	0.4383
(16, 24] - MHA & FFN	0.7189	0.3598	0.1711	0.4941	0.2340	0.3513	0.4166
(12, 20] - MHA & FFN	0.7233	0.3714	0.1887	0.5035	0.2494	0.3614	0.4278
(10, 18] - MHA & FFN	0.7257	0.3781	0.1935	0.5097	0.2548	0.3697	0.4324
(8, 16] - FFN	0.7264	0.3770	0.1957	0.5114	0.2495	0.3701	0.4330
(8, 16] - MHA	0.7189	0.3631	0.1802	0.4970	0.2371	0.3553	0.4207
(8, 16] - down_proj	0.7215	0.3704	0.1845	0.5037	0.2438	0.3637	0.4255
(8, 16] - up_proj	0.7193	0.3644	0.1797	0.4985	0.2384	0.3564	0.4211

Table 13: The detailed speciality performance of Law task under the 7B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 16] - MHA & FFN	0.4847	0.3189	0.4225	0.2931	0.3052	0.4064	0.6453	0.4087
(8, 24] - MHA & FFN	0.4943	0.3488	0.4460	0.3147	0.3205	0.4174	0.7314	0.4297
(16, 32] - MHA & FFN	0.5448	0.3588	0.4533	0.2992	0.3247	0.4416	0.7475	0.4523
(4, 20] - MHA & FFN	0.4902	0.3240	0.4368	0.3013	0.3138	0.4109	0.7217	0.4170
(8, 16] - MHA & FFN	0.5454	0.3512	0.4487	0.3133	0.3236	0.4224	0.7354	0.4484
(16, 24] - MHA & FFN	0.5791	0.3591	0.4578	0.3019	0.3327	0.4471	0.7495	0.4653
(12, 20] - MHA & FFN	0.5494	0.3545	0.4524	0.3148	0.3284	0.4252	0.7413	0.4521
(10, 18] - MHA & FFN	0.5463	0.3525	0.4496	0.3116	0.3239	0.4206	0.7425	0.4495
(8, 16] - FFN	0.5861	0.3525	0.4499	0.3225	0.3096	0.4279	0.7396	0.4628
(8, 16] - MHA	0.6037	0.3487	0.4513	0.3230	0.3108	0.4281	0.7434	0.4679
(8, 16] - down_proj	0.5957	0.3503	0.4502	0.3126	0.3211	0.4263	0.7408	0.4654
(8, 16] - up_proj	0.6046	0.3544	0.4515	0.3295	0.3034	0.4255	0.7475	0.4702

Table 14: The detailed versatility performance of Law task under the 7B model in different exploring layer ranges and modules.

	Accuracy
(0, 16] - MHA & FFN	0.084
(8, 24] - MHA & FFN	0.088
(16, 32] - MHA & FFN	0.064
(4, 20] - MHA & FFN	0.088
(8, 16] - MHA & FFN	0.084
(16, 24] - MHA & FFN	0.056
(12, 20] - MHA & FFN	0.072
(10, 18] - MHA & FFN	0.076
(8, 16] - FFN	0.080
(8, 16] - MHA	0.056
(8, 16] - down_proj	0.072
(8, 16] - up_proj	0.064

Table 15: The detailed speciality performance of Math task under the 7B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 16] - MHA & FFN	0.5554	0.3282	0.4312	0.3029	0.2917	0.4005	0.7296	0.4383
(8, 24] - MHA & FFN	0.5617	0.3539	0.4535	0.3008	0.3304	0.4423	0.7408	0.4564
(16, 32] - MHA & FFN	0.6043	0.3611	0.4616	0.3054	0.3449	0.4507	0.7455	0.4756
(4, 20] - MHA & FFN	0.5579	0.3352	0.4331	0.3085	0.2964	0.3959	0.7314	0.4421
(8, 16] - MHA & FFN	0.5958	0.3556	0.4551	0.3194	0.3235	0.4341	0.7435	0.4688
(16, 24] - MHA & FFN	0.6141	0.3629	0.4646	0.3108	0.3486	0.4537	0.7452	0.4805
(12, 20] - MHA & FFN	0.6026	0.3565	0.4554	0.3069	0.3297	0.4416	0.7435	0.4715
(10, 18] - MHA & FFN	0.5988	0.3558	0.4558	0.3209	0.3197	0.4412	0.7416	0.4701
(8, 16] - FFN	0.6151	0.3563	0.4581	0.3168	0.3280	0.4394	0.7483	0.4765
(8, 16] - MHA	0.6245	0.3571	0.4609	0.3240	0.3221	0.4532	0.7447	0.4808
(8, 16] - down_proj	0.6205	0.3548	0.4568	0.3132	0.3228	0.4466	0.7441	0.4774
(8, 16] - up_proj	0.6255	0.3564	0.4607	0.3199	0.3297	0.4501	0.7438	0.4809

Table 16: The detailed versatility performance of Math task under the 7B model in different exploring layer ranges and modules.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
(0, 16] - MHA & FFN	0.7381	0.4794	0.3473	0.5481	0.3987	0.4913	0.5216
(8, 24] - MHA & FFN	0.7395	0.4849	0.3548	0.5517	0.4051	0.4978	0.5264
(16, 32] - MHA & FFN	0.7277	0.4604	0.3252	0.5248	0.3803	0.4762	0.5044
(4, 20] - MHA & FFN	0.7376	0.4856	0.3555	0.5519	0.4064	0.4986	0.5262
(8, 16] - MHA & FFN	0.7369	0.4776	0.3457	0.5455	0.3973	0.4901	0.5200
(16, 24] - MHA & FFN	0.7261	0.4594	0.3286	0.5234	0.3795	0.4753	0.5047
(12, 20] - MHA & FFN	0.7304	0.4685	0.3353	0.5376	0.3883	0.4792	0.5114
(10, 18] - MHA & FFN	0.7332	0.4731	0.3398	0.5412	0.3926	0.4854	0.5153
(8, 16] - FFN	0.7337	0.4714	0.3425	0.5402	0.3901	0.4837	0.5159
(8, 16] - MHA	0.7263	0.4575	0.3264	0.5274	0.3757	0.4693	0.5034
(8, 16] - down_proj	0.7291	0.4643	0.3311	0.5342	0.3828	0.4761	0.5082
(8, 16] - up_proj	0.7279	0.4604	0.3298	0.5313	0.3789	0.4711	0.5061

Table 17: The detailed speciality performance of CAG task under the 7B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 16] - MHA & FFN	0.5508	0.3124	0.4263	0.2992	0.3071	0.4276	0.6713	0.4298
(8, 24] - MHA & FFN	0.5555	0.3497	0.4449	0.3037	0.3158	0.4308	0.7294	0.4500
(16, 32] - MHA & FFN	0.5707	0.3576	0.4538	0.3068	0.3254	0.4354	0.7475	0.4607
(4, 20] - MHA & FFN	0.5532	0.3404	0.4366	0.3019	0.3059	0.4233	0.7154	0.4434
(8, 16] - MHA & FFN	0.5846	0.3508	0.4462	0.3058	0.3121	0.4294	0.7374	0.4605
(16, 24] - MHA & FFN	0.5954	0.3589	0.4523	0.3085	0.3201	0.4411	0.7396	0.4689
(12, 20] - MHA & FFN	0.5881	0.3515	0.4467	0.3091	0.3157	0.4265	0.7353	0.4621
(10, 18] - MHA & FFN	0.5865	0.3521	0.4454	0.3023	0.3108	0.4331	0.7355	0.4613
(8, 16] - FFN	0.6055	0.3519	0.4477	0.3006	0.3221	0.4295	0.7385	0.4684
(8, 16] - MHA	0.6136	0.3492	0.4506	0.3011	0.3282	0.4275	0.7452	0.4711
(8, 16] - down_proj	0.6143	0.3516	0.4484	0.3068	0.3184	0.4240	0.7447	0.4714
(8, 16] - up_proj	0.6201	0.3533	0.4515	0.3116	0.3244	0.4271	0.7431	0.4750

Table 18: The detailed versatility performance of CAG task under the 7B model in different exploring layer ranges and modules.

A.12.2 13B Results

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
(0, 20] - MHA & FFN	0.7418	0.4045	0.2204	0.5525	0.2823	0.3786	0.4555
(10, 30] - MHA & FFN	0.7417	0.4060	0.2223	0.5537	0.284	0.3802	0.4567
(20, 40] - MHA & FFN	0.7225	0.3683	0.1802	0.5213	0.2424	0.3413	0.4237
(5, 25] - MHA & FFN	0.7416	0.4071	0.2238	0.5533	0.2846	0.3829	0.4574
(10, 20] - MHA & FFN	0.7341	0.3893	0.2056	0.5383	0.2651	0.3645	0.4430
(20, 30] - MHA & FFN	0.7203	0.3683	0.1756	0.5192	0.2444	0.3415	0.4214
(15, 25] - MHA & FFN	0.7258	0.3771	0.1869	0.5278	0.2520	0.3517	0.4299
(13, 23] - MHA & FFN	0.7284	0.3820	0.1928	0.5328	0.2571	0.3562	0.4343
(10, 20] - FFN	0.7291	0.3801	0.1965	0.5301	0.2565	0.3537	0.4352
(10, 20] - MHA	0.7225	0.3619	0.1691	0.5136	0.2372	0.3350	0.4178
(10, 20] - down_proj	0.7242	0.3671	0.1802	0.5193	0.2418	0.3402	0.4238
(10, 20] - up_proj	0.7206	0.3635	0.1710	0.5156	0.2367	0.3381	0.4183

Table 19: The detailed speciality performance of Finance task under the 13B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 20] - MHA & FFN	0.4511	0.3582	0.4584	0.3023	0.3386	0.5133	0.6793	0.4226
(10, 30] - MHA & FFN	0.4643	0.3993	0.4901	0.3235	0.3946	0.5328	0.7095	0.4512
(20, 40] - MHA & FFN	0.5215	0.4065	0.5008	0.3581	0.4179	0.5141	0.7137	0.4762
(5, 25] - MHA & FFN	0.4562	0.3839	0.4687	0.3132	0.3801	0.5042	0.6772	0.4363
(10, 20] - MHA & FFN	0.5238	0.4007	0.4935	0.3230	0.4093	0.5454	0.6964	0.4727
(20, 30] - MHA & FFN	0.5615	0.4071	0.5001	0.3156	0.4021	0.5875	0.6951	0.4896
(15, 25] - MHA & FFN	0.5435	0.4048	0.4945	0.3388	0.4185	0.5113	0.7097	0.4810
(13, 23] - MHA & FFN	0.5361	0.4023	0.4932	0.3349	0.4015	0.5388	0.6973	0.4772
(10, 20] - FFN	0.5750	0.4009	0.4975	0.3301	0.4004	0.5227	0.7369	0.4912
(10, 20] - MHA	0.6067	0.4012	0.4967	0.3268	0.4026	0.5232	0.7355	0.5015
(10, 20] - down_proj	0.5925	0.4023	0.4988	0.3287	0.4078	0.5193	0.7517	0.4979
(10, 20] - up_proj	0.6078	0.4045	0.4960	0.3355	0.4001	0.5339	0.7168	0.5028

Table 20: The detailed versatility performance of Finance task under the 13B model in different exploring layer ranges and modules.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
(0, 20] - MHA & FFN	0.7472	0.4341	0.2261	0.5585	0.3142	0.4297	0.4691
(10, 30] - MHA & FFN	0.7486	0.4363	0.2284	0.5594	0.3171	0.4325	0.4711
(20, 40] - MHA & FFN	0.7253	0.3946	0.1848	0.5231	0.2717	0.3890	0.4349
(5, 25] - MHA & FFN	0.7459	0.4354	0.2303	0.5577	0.3171	0.4315	0.4705
(10, 20] - MHA & FFN	0.7404	0.4201	0.2102	0.5458	0.2995	0.4151	0.4569
(20, 30] - MHA & FFN	0.7318	0.3899	0.1748	0.5210	0.2645	0.3843	0.4322
(15, 25] - MHA & FFN	0.7304	0.4011	0.1868	0.5313	0.2756	0.3965	0.4395
(13, 23] - MHA & FFN	0.7362	0.4127	0.1998	0.539	0.2898	0.4094	0.4496
(10, 20] - FFN	0.7360	0.4115	0.2062	0.5351	0.2906	0.4089	0.4512
(10, 20] - MHA	0.7255	0.3896	0.1764	0.5168	0.2663	0.3858	0.4304
(10, 20] - down_proj	0.7313	0.3961	0.1878	0.5239	0.2741	0.3901	0.4383
(10, 20] - up_proj	0.7242	0.3896	0.1773	0.5174	0.2679	0.3836	0.4303

Table 21: The detailed speciality performance of Law task under the 13B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 20] - MHA & FFN	0.4405	0.3614	0.4511	0.3116	0.3606	0.4666	0.6653	0.4176
(10, 30] - MHA & FFN	0.4608	0.3943	0.4874	0.3329	0.4150	0.5273	0.6758	0.4462
(20, 40] - MHA & FFN	0.5074	0.4065	0.5014	0.3551	0.4147	0.5441	0.6914	0.4718
(5, 25] - MHA & FFN	0.4463	0.3757	0.4646	0.3132	0.3939	0.4742	0.6775	0.4289
(10, 20] - MHA & FFN	0.5006	0.4006	0.4934	0.3547	0.4103	0.5092	0.6993	0.4649
(20, 30] - MHA & FFN	0.5474	0.4052	0.5015	0.3690	0.4122	0.5213	0.7034	0.4847
(15, 25] - MHA & FFN	0.5263	0.4019	0.4951	0.3504	0.4084	0.5263	0.6956	0.4744
(13, 23] - MHA & FFN	0.5066	0.4004	0.4925	0.3509	0.4021	0.5218	0.6953	0.4688
(10, 20] - FFN	0.5601	0.4013	0.4952	0.3302	0.4185	0.5203	0.7114	0.4855
(10, 20] - MHA	0.5983	0.4039	0.4966	0.3210	0.4126	0.5205	0.7323	0.4995
(10, 20] - down_proj	0.5805	0.4025	0.4974	0.3257	0.4166	0.5219	0.7255	0.4935
(10, 20] - up_proj	0.6025	0.4017	0.4993	0.3318	0.4197	0.5244	0.7218	0.5012

Table 22: The detailed versatility performance of Law task under the 13B model in different exploring layer ranges and modules.

	Accuracy
(0, 20] - MHA & FFN	0.112
(10, 30] - MHA & FFN	0.120
(20, 40] - MHA & FFN	0.068
(5, 25] - MHA & FFN	0.116
(10, 20] - MHA & FFN	0.112
(20, 30] - MHA & FFN	0.056
(15, 25] - MHA & FFN	0.076
(13, 23] - MHA & FFN	0.100
(10, 20] - FFN	0.108
(10, 20] - MHA	0.064
(10, 20] - down_proj	0.076
(10, 20] - up_proj	0.068

Table 23: The detailed speciality performance of Math task under the 13B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 20] - MHA & FFN	0.5793	0.3619	0.4528	0.3101	0.3493	0.4286	0.7234	0.4647
(10, 30] - MHA & FFN	0.5879	0.3936	0.5011	0.3602	0.4021	0.5208	0.7215	0.4942
(20, 40] - MHA & FFN	0.6090	0.4017	0.5115	0.3683	0.4158	0.5305	0.7317	0.5074
(5, 25] - MHA & FFN	0.5838	0.3875	0.4718	0.3294	0.3933	0.4311	0.7331	0.4810
(10, 20] - MHA & FFN	0.6107	0.3971	0.5049	0.3535	0.4098	0.5188	0.7375	0.5043
(20, 30] - MHA & FFN	0.6202	0.4032	0.5141	0.3705	0.4179	0.5273	0.7404	0.5125
(15, 25] - MHA & FFN	0.6175	0.4003	0.5066	0.3558	0.4095	0.5057	0.7558	0.5081
(13, 23] - MHA & FFN	0.6135	0.4084	0.5053	0.3455	0.4122	0.5318	0.7312	0.5091
(10, 20] - FFN	0.6208	0.4009	0.5091	0.3572	0.4178	0.5174	0.7441	0.5102
(10, 20] - MHA	0.6274	0.4013	0.5087	0.3597	0.4141	0.5065	0.7505	0.5125
(10, 20] - down_proj	0.6259	0.4002	0.5076	0.3528	0.4197	0.5043	0.7533	0.5112
(10, 20] - up_proj	0.6307	0.4015	0.5095	0.3659	0.4166	0.5041	0.7518	0.5139

Table 24: The detailed versatility performance of Math task under the 13B model in different exploring layer ranges and modules.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
(0, 20] - MHA & FFN	0.7487	0.5133	0.3845	0.5791	0.4313	0.5294	0.5488
(10, 30] - MHA & FFN	0.7493	0.5185	0.3869	0.5839	0.4369	0.5345	0.5515
(20, 40] - MHA & FFN	0.7381	0.4926	0.3604	0.5561	0.4108	0.5109	0.5303
(5, 25] - MHA & FFN	0.7479	0.5173	0.3881	0.5821	0.4364	0.5333	0.5510
(10, 20] - MHA & FFN	0.7471	0.5106	0.3761	0.5769	0.4286	0.5264	0.5446
(20, 30] - MHA & FFN	0.7379	0.4885	0.3542	0.5542	0.4013	0.5102	0.5268
(15, 25] - MHA & FFN	0.7284	0.4992	0.3645	0.5655	0.4173	0.5149	0.5307
(13, 23] - MHA & FFN	0.7446	0.5037	0.3702	0.5699	0.4203	0.5208	0.5394
(10, 20] - FFN	0.7449	0.5035	0.3705	0.5711	0.4207	0.5188	0.5397
(10, 20] - MHA	0.7372	0.4898	0.3534	0.5587	0.4071	0.5036	0.5267
(10, 20] - down_proj	0.7398	0.4964	0.3601	0.5655	0.4142	0.5094	0.5321
(10, 20] - up_proj	0.7372	0.4921	0.3575	0.5627	0.4102	0.5035	0.5289

Table 25: The detailed speciality performance of CAG task under the 13B model in different exploring layer ranges and modules.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
(0, 20] - MHA & FFN	0.5543	0.3621	0.4556	0.3062	0.3439	0.4709	0.7014	0.4573
(10, 30] - MHA & FFN	0.5641	0.3963	0.4956	0.3424	0.3988	0.5253	0.7159	0.4852
(20, 40] - MHA & FFN	0.5852	0.4041	0.5062	0.3632	0.4168	0.5223	0.7224	0.4984
(5, 25] - MHA & FFN	0.5574	0.3857	0.4702	0.3213	0.3867	0.4676	0.7053	0.4711
(10, 20] - MHA & FFN	0.5905	0.3984	0.4987	0.3387	0.4101	0.5306	0.7155	0.4959
(20, 30] - MHA & FFN	0.6037	0.4051	0.5071	0.3431	0.4203	0.5486	0.7168	0.5052
(15, 25] - MHA & FFN	0.5962	0.4028	0.5006	0.3473	0.4140	0.5085	0.7325	0.4998
(13, 23] - MHA & FFN	0.5920	0.4009	0.4992	0.3403	0.4068	0.5353	0.7142	0.4973
(10, 20] - FFN	0.6142	0.4009	0.5005	0.3511	0.4008	0.5237	0.7265	0.5054
(10, 20] - MHA	0.6224	0.3976	0.5032	0.3437	0.4093	0.5158	0.7440	0.5077
(10, 20] - down_proj	0.6243	0.4012	0.5015	0.3387	0.4098	0.5074	0.7483	0.5088
(10, 20] - up_proj	0.6286	0.4030	0.5041	0.3524	0.4094	0.5191	0.7352	0.5119

Table 26: The detailed versatility performance of CAG task under the 13B model in different exploring layer ranges and modules.

A.13 Optional Solution

	7B				13B			
	Finance	Law	Math	CAG	Finance	Law	Math	CAG
MHA & FFN	**0.8241**	**0.8382**	**0.4893**	**0.9185**	**0.8901**	**0.9039**	**0.5632**	**0.9931**
FFN	0.8204	0.8342	0.4871	0.9156	0.8842	0.8994	0.5630	0.9904
MHA	0.8066	0.8207	0.4650	0.9033	0.8667	0.8807	0.5190	0.9771
down_proj	0.8138	0.8258	0.4778	0.9082	0.8743	0.8882	0.5299	0.9833
up_proj	0.8125	0.8241	0.4726	0.9085	0.8686	0.8808	0.5235	0.9824

Table 27: The *Uni. w/o instruct* scores in various trainable modules within the $(N \times 25\%, N \times 50\%]$ layer range of the 7B and 13B models.

In our context, instruction following is a crucial aspect of LLM's versatility. However, in scenarios where users can provide clear and concise instructions, the model's precision in this aspect may be less critical. We conduct additional analysis on the results of LLM's versatility without instruction following (*Vers. w/o instruct*), considering different trainable modules in the layer range $(N \times 25\%, N \times 50\%]$ under 7B and 13B models. The derived Uni. score without instruction following (*Uni. w/o instruct*) shows that "$(N \times 25\%, N \times 50\%]$ - MHA & FFN" generally outperforms "$(N \times 25\%, N \times 50\%]$ - FFN". This suggests that "$(N \times 25\%, N \times 50\%]$ - MHA & FFN" could be a viable alternative when users value instruction following aspect less.

A.14 Detailed Results of *CoFiTune* and Baselines under BLOOMZ Backbone

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6814	0.2869	0.1042	0.4177	0.1639	0.2792	0.3575
Full SFT	0.7106	0.3581	0.1893	0.4985	0.2292	0.3468	0.4193
LoRA	0.6932	0.3167	0.1406	0.4536	0.1944	0.3023	0.3835
Wise-FT	0.7015	0.3296	0.1614	0.4651	0.2047	0.3192	0.3975
V-SoftMask	0.7113	0.3594	0.1885	0.5002	0.2295	0.3485	0.4197
L1	0.7007	0.3289	0.1572	0.4685	0.2036	0.3161	0.3956
L2	0.7043	0.3385	0.1651	0.4769	0.2114	0.3273	0.4026
CoFiTune	0.7035	0.3376	0.1635	0.4742	0.2128	0.3258	0.4015

Table 28: The detailed speciality performance of Finance task under the 7B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6109	0.3457	0.4232	0.2831	0.2947	0.4168	0.6984	0.4599
Full SFT	0.4471	0.2943	0.3787	0.2518	0.2563	0.3742	0.6325	0.3733
LoRA	0.5968	0.2965	0.3781	0.2573	0.2604	0.3216	0.6728	0.4238
Wise-FT	0.5884	0.3216	0.4028	0.2669	0.2771	0.3865	0.6806	0.4376
V-SoftMask	0.4508	0.2981	0.3839	0.2562	0.2675	0.3804	0.6313	0.3776
L1	0.5562	0.3015	0.3987	0.2604	0.2698	0.3967	0.6681	0.4188
L2	0.5136	0.2973	0.3863	0.2615	0.2677	0.4023	0.6538	0.3991
CoFiTune	0.5927	0.3341	0.4075	0.2712	0.2789	0.3905	0.6894	0.4448

Table 29: The detailed versatility performance of Finance task under the 7B model in different methods.

A.15 Detailed Results of *CoFiTune* and Baselines under Chinese-Alpaca-Pro Backbone

A.15.1 7B Results

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6912	0.3038	0.1189	0.4384	0.1766	0.2963	0.3713
Full SFT	0.7376	0.3973	0.2163	0.5440	0.2752	0.3728	0.4504
LoRA	0.7159	0.3487	0.1527	0.5047	0.2233	0.3182	0.4058
Wise-FT	0.7208	0.3597	0.1764	0.5134	0.2331	0.3327	0.4189
V-SoftMask	0.7372	0.3980	0.2175	0.5456	0.2764	0.3720	0.4509
L1	0.7181	0.3551	0.1737	0.5091	0.2282	0.3277	0.4156
L2	0.7254	0.3649	0.1783	0.5192	0.2388	0.3368	0.4229
CoFiTune	0.7222	0.3645	0.1788	0.5197	0.2374	0.3365	0.4217

Table 30: The detailed speciality performance of Finance task under the 7B model across different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6237	0.3671	0.4639	0.3101	0.3379	0.4622	0.7455	0.4849
Full SFT	0.4608	0.3158	0.4188	0.2666	0.2969	0.4441	0.6673	0.3984
LoRA	0.6074	0.3145	0.4211	0.2738	0.2832	0.4039	0.7234	0.4476
Wise-FT	0.5979	0.3436	0.4424	0.2828	0.3014	0.4542	0.7314	0.4613
V-SoftMask	0.4643	0.3204	0.4237	0.2708	0.3015	0.4467	0.6759	0.4028
L1	0.5767	0.3213	0.4293	0.2733	0.2916	0.4376	0.7148	0.4424
L2	0.5256	0.3179	0.4304	0.2762	0.2927	0.4262	0.7261	0.4246
CoFiTune	0.6018	0.3566	0.4465	0.2811	0.3099	0.4479	0.7474	0.4683

Table 31: The detailed versatility performance of Finance task under the 7B model across different methods.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6934	0.3083	0.1247	0.4375	0.1818	0.3057	0.3755
Full SFT	0.7407	0.4039	0.2282	0.5329	0.2754	0.4036	0.4576
LoRA	0.7195	0.3594	0.1725	0.4926	0.2335	0.3521	0.4171
Wise-FT	0.7266	0.3751	0.1944	0.5093	0.2477	0.3684	0.4320
V-SoftMask	0.7402	0.4046	0.2289	0.5365	0.2769	0.4005	0.4579
L1	0.7243	0.3695	0.1872	0.5046	0.2443	0.3595	0.4270
L2	0.7315	0.3756	0.1953	0.5103	0.2467	0.3699	0.4341
CoFiTune	0.7271	0.3772	0.1942	0.5119	0.2491	0.3706	0.4328

Table 32: The detailed speciality performance of Law task under the 7B model across different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6237	0.3671	0.4639	0.3101	0.3379	0.4622	0.7455	0.4849
Full SFT	0.4280	0.3162	0.4212	0.3039	0.3046	0.3941	0.6824	0.3884
LoRA	0.6151	0.3233	0.4335	0.3178	0.3109	0.3578	0.7475	0.4573
Wise-FT	0.5925	0.3335	0.4475	0.3006	0.3222	0.4257	0.7415	0.4578
V-SoftMask	0.4321	0.3209	0.4277	0.3101	0.3142	0.3982	0.6885	0.3936
L1	0.5657	0.3298	0.4378	0.3177	0.3193	0.4133	0.7011	0.4444
L2	0.5243	0.3246	0.4335	0.3058	0.3269	0.4065	0.6946	0.4275
CoFiTune	0.5893	0.3554	0.4551	0.3295	0.3104	0.4321	0.7482	0.4665

Table 33: The detailed versatility performance of Law task under the 7B model across different methods.

	Accuracy
ZeroShot	0.048
Full SFT	0.100
LoRA	0.056
Wise-FT	0.064
V-SoftMask	0.096
L1	0.068
L2	0.076
CoFiTune	0.080

Table 34: The detailed speciality performance of Math task under the 7B model across different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6237	0.3671	0.4639	0.3101	0.3379	0.4622	0.7455	0.4849
Full SFT	0.4874	0.3227	0.4450	0.3194	0.3071	0.4421	0.7114	0.4184
LoRA	0.6077	0.3133	0.4326	0.2899	0.2782	0.4311	0.7315	0.4512
Wise-FT	0.6156	0.3515	0.4515	0.3116	0.3486	0.3965	0.7495	0.4729
V-SoftMask	0.4921	0.3267	0.4496	0.3218	0.3105	0.4453	0.7213	0.4226
L1	0.5888	0.3273	0.4464	0.3153	0.3067	0.4351	0.7285	0.4542
L2	0.5464	0.3215	0.4447	0.3083	0.3113	0.4291	0.7301	0.4375
CoFiTune	0.6186	0.3589	0.4616	0.3221	0.3291	0.4436	0.7516	0.4797

Table 35: The detailed versatility performance of Math task under the 7B model across different methods.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.7012	0.4056	0.2712	0.4559	0.3241	0.4369	0.4593
Full SFT	0.7486	0.5005	0.3703	0.5632	0.4231	0.5153	0.5398
LoRA	0.7314	0.4632	0.3308	0.5334	0.3796	0.4765	0.5085
Wise-FT	0.7323	0.4664	0.3355	0.5367	0.3833	0.4791	0.5114
V-SoftMask	0.7483	0.5004	0.3714	0.5618	0.4232	0.5162	0.5400
L1	0.7324	0.4660	0.3276	0.5389	0.3824	0.4767	0.5087
L2	0.7355	0.4713	0.3407	0.5411	0.3913	0.4815	0.5158
CoFiTune	0.7356	0.4720	0.3422	0.5414	0.3905	0.4840	0.5166

Table 36: The detailed speciality performance of CAG task under the 7B model across different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6237	0.3671	0.4639	0.3101	0.3379	0.4622	0.7455	0.4849
Full SFT	0.4517	0.3111	0.4123	0.2822	0.3021	0.4035	0.6613	0.3917
LoRA	0.6178	0.3215	0.4319	0.2945	0.3058	0.3819	0.7455	0.4571
Wise-FT	0.6009	0.3423	0.4460	0.2930	0.3336	0.4321	0.7255	0.4631
V-SoftMask	0.4549	0.3165	0.4172	0.2853	0.3102	0.4040	0.6693	0.3962
L1	0.5746	0.3283	0.4334	0.3012	0.3153	0.4137	0.7033	0.4454
L2	0.5205	0.3252	0.4307	0.3034	0.3128	0.4173	0.6894	0.4255
CoFiTune	0.6082	0.3531	0.4535	0.3078	0.3257	0.4381	0.7425	0.4716

Table 37: The detailed versatility performance of CAG task under the 7B model across different methods.

A.15.2 13B Results

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6934	0.3124	0.1239	0.4457	0.1881	0.3035	0.3766
Full SFT	0.7475	0.4192	0.2415	0.5614	0.2997	0.3964	0.4761
LoRA	0.7217	0.3628	0.1692	0.5209	0.2365	0.3311	0.4179
Wise-FT	0.7258	0.3729	0.1936	0.5226	0.2478	0.3485	0.4308
V-SoftMask	0.7477	0.4172	0.2408	0.5595	0.2978	0.3943	0.4752
L1	0.7265	0.3741	0.1857	0.5242	0.2486	0.3494	0.4287
L2	0.7302	0.3817	0.1985	0.5312	0.2574	0.3566	0.4368
CoFiTune	0.7295	0.3799	0.1958	0.5294	0.2561	0.3542	0.4351

Table 38: The detailed speciality performance of Finance task under the 13B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6320	0.4057	0.5224	0.3752	0.4141	0.5448	0.7555	0.5201
Full SFT	0.4042	0.3555	0.4431	0.3333	0.3228	0.4652	0.6513	0.4009
LoRA	0.5986	0.3721	0.4704	0.3147	0.3486	0.4867	0.7314	0.4804
Wise-FT	0.5777	0.3908	0.4873	0.3240	0.3801	0.5318	0.7136	0.4853
V-SoftMask	0.4082	0.3593	0.4484	0.3311	0.3279	0.4742	0.6603	0.4055
L1	0.5577	0.3779	0.4742	0.3214	0.3564	0.4902	0.7288	0.4701
L2	0.4786	0.3723	0.4702	0.3135	0.3464	0.5015	0.7192	0.4404
CoFiTune	0.5779	0.4037	0.5019	0.3375	0.4034	0.5244	0.7423	0.4945

Table 39: The detailed versatility performance of Finance task under the 13B model in different methods.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6950	0.3197	0.1208	0.4473	0.1959	0.3156	0.3785
Full SFT	0.7559	0.4454	0.2471	0.5627	0.3257	0.4479	0.4894
LoRA	0.7287	0.3820	0.1936	0.5167	0.2556	0.3736	0.4347
Wise-FT	0.7294	0.3975	0.2006	0.5245	0.2741	0.3938	0.4425
V-SoftMask	0.7551	0.4474	0.2448	0.5652	0.3295	0.4477	0.4890
L1	0.7272	0.3902	0.1956	0.5204	0.2652	0.3849	0.4376
L2	0.7313	0.4029	0.2059	0.5295	0.2804	0.3986	0.4467
CoFiTune	0.7354	0.4110	0.2046	0.5357	0.2898	0.4075	0.4503

Table 40: The detailed speciality performance of Law task under the 13B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6320	0.4057	0.5224	0.3752	0.4141	0.5448	0.7555	0.5201
Full SFT	0.4004	0.3655	0.4361	0.3116	0.3486	0.4311	0.6533	0.4007
LoRA	0.6089	0.3754	0.4794	0.3242	0.3878	0.5243	0.6814	0.4879
Wise-FT	0.5688	0.4042	0.4934	0.3364	0.3820	0.5278	0.7275	0.4888
V-SoftMask	0.4065	0.3706	0.4424	0.3163	0.3323	0.4737	0.6473	0.4065
L1	0.5553	0.3838	0.4803	0.3342	0.3874	0.5134	0.6864	0.4732
L2	0.4926	0.3795	0.4759	0.3303	0.3787	0.5118	0.6826	0.4493
CoFiTune	0.5646	0.4068	0.5003	0.3318	0.4158	0.5303	0.7234	0.4903

Table 41: The detailed versatility performance of Law task under the 13B model in different methods.

	Accuracy
ZeroShot	0.076
Full SFT	0.140
LoRA	0.084
Wise-FT	0.100
V-SoftMask	0.140
L1	0.092
L2	0.108
CoFiTune	0.112

Table 42: The detailed speciality performance of Math task under the 13B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6320	0.4057	0.5224	0.3752	0.4141	0.5448	0.7555	0.5201
Full SFT	0.5409	0.3608	0.4521	0.3318	0.3606	0.4105	0.7054	0.4512
LoRA	0.6096	0.3696	0.4574	0.3225	0.3524	0.4271	0.7273	0.4788
Wise-FT	0.6257	0.3959	0.4993	0.3612	0.4216	0.4812	0.7335	0.5070
V-SoftMask	0.5447	0.3652	0.4565	0.3385	0.3632	0.4131	0.7112	0.4554
L1	0.6013	0.3766	0.4659	0.3408	0.3664	0.4282	0.7286	0.4813
L2	0.5845	0.3721	0.4617	0.3358	0.3623	0.4252	0.7233	0.4728
CoFiTune	0.6231	0.4035	0.5123	0.3616	0.4158	0.5193	0.7527	0.5130

Table 43: The detailed versatility performance of Math task under the 13B model in different methods.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.7044	0.4124	0.2743	0.4619	0.3324	0.4429	0.4637
Full SFT	0.7594	0.5294	0.3953	0.5921	0.4522	0.5438	0.5613
LoRA	0.7411	0.4984	0.3662	0.5655	0.4151	0.5147	0.5352
Wise-FT	0.7401	0.4961	0.3621	0.5634	0.4146	0.5103	0.5327
V-SoftMask	0.7592	0.5295	0.3955	0.5933	0.4510	0.5442	0.5614
L1	0.7393	0.4921	0.3593	0.5603	0.4102	0.5058	0.5302
L2	0.7438	0.5018	0.3682	0.5685	0.4199	0.5168	0.5379
CoFiTune	0.7452	0.5041	0.3729	0.5705	0.4224	0.5195	0.5406

Table 44: The detailed speciality performance of CAG task under the 13B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6320	0.4057	0.5224	0.3752	0.4141	0.5448	0.7555	0.5201
Full SFT	0.5046	0.3638	0.4476	0.3325	0.3417	0.4378	0.6783	0.4386
LoRA	0.6241	0.3708	0.4639	0.3186	0.3505	0.4569	0.7294	0.4863
Wise-FT	0.6082	0.3934	0.4933	0.3426	0.4009	0.5065	0.7236	0.4983
V-SoftMask	0.5102	0.3685	0.4541	0.3363	0.3471	0.4451	0.6874	0.4443
L1	0.5873	0.3772	0.4702	0.3311	0.3614	0.4592	0.7286	0.4782
L2	0.5512	0.3723	0.4659	0.3246	0.3544	0.4634	0.7213	0.4631
CoFiTune	0.6183	0.4056	0.5066	0.3481	0.4107	0.5303	0.7375	0.5105

Table 45: The detailed versatility performance of CAG task under the 13B model in different methods.

A.15.3 33B Results

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6961	0.3192	0.1282	0.4533	0.1938	0.3108	0.3811
Full SFT	0.7516	0.4260	0.2534	0.5666	0.3075	0.4041	0.4769
LoRA	0.7235	0.3656	0.1718	0.5203	0.2419	0.3347	0.4207
Wise-FT	0.7277	0.3759	0.1919	0.5282	0.2511	0.3484	0.4318
V-SoftMask	0.7513	0.4267	0.2538	0.5661	0.3088	0.4051	0.4772
L1	0.7265	0.3729	0.1868	0.5265	0.2464	0.3459	0.4287
L2	0.7298	0.3816	0.1975	0.5344	0.2553	0.3552	0.4363
CoFiTune	0.7331	0.3861	0.2066	0.5403	0.2605	0.3573	0.4419

Table 46: The detailed speciality performance of Finance task under the 33B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6386	0.4684	0.5592	0.3953	0.4732	0.5744	0.7936	0.5553
Full SFT	0.4426	0.4176	0.4975	0.3333	0.4103	0.5086	0.7375	0.4525
LoRA	0.6041	0.4251	0.5192	0.3767	0.4097	0.5228	0.7673	0.5161
Wise-FT	0.5989	0.4513	0.5328	0.3776	0.4514	0.5469	0.7555	0.5277
V-SoftMask	0.4484	0.4215	0.5014	0.3294	0.4184	0.5172	0.7404	0.4571
L1	0.5706	0.4273	0.5063	0.3373	0.4236	0.5161	0.7483	0.5014
L2	0.5181	0.4225	0.5014	0.3294	0.4185	0.5175	0.7404	0.4807
CoFiTune	0.5918	0.4634	0.5358	0.3922	0.4601	0.4992	0.7916	0.5303

Table 47: The detailed versatility performance of Finance task under the 33B model in different methods.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.6962	0.3263	0.1355	0.4576	0.1991	0.3221	0.3860
Full SFT	0.7604	0.4578	0.2601	0.5716	0.3391	0.4629	0.4927
LoRA	0.7308	0.3920	0.2082	0.5246	0.2642	0.3872	0.4436
Wise-FT	0.7346	0.4017	0.2295	0.5338	0.2749	0.3965	0.4552
V-SoftMask	0.7602	0.4587	0.2605	0.5731	0.3396	0.4634	0.4931
L1	0.7313	0.3962	0.2239	0.5295	0.2688	0.3903	0.4504
L2	0.7384	0.4095	0.2378	0.5402	0.2831	0.4054	0.4619
CoFiTune	0.7392	0.4122	0.2369	0.5443	0.2843	0.4075	0.4627

Table 48: The detailed speciality performance of Law task under the 33B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6386	0.4684	0.5592	0.3953	0.4732	0.5744	0.7936	0.5553
Full SFT	0.4381	0.4089	0.4926	0.3350	0.4147	0.5143	0.7074	0.4465
LoRA	0.6074	0.4385	0.5138	0.3567	0.4173	0.5259	0.7555	0.5199
Wise-FT	0.5950	0.4519	0.5309	0.3631	0.4576	0.5554	0.7475	0.5259
V-SoftMask	0.4438	0.4142	0.4982	0.3373	0.4164	0.5189	0.7192	0.4522
L1	0.5719	0.4346	0.5024	0.3432	0.4165	0.5236	0.7255	0.5032
L2	0.5042	0.4283	0.4981	0.3385	0.4096	0.5238	0.7203	0.4769
CoFiTune	0.5874	0.4625	0.5352	0.3938	0.4663	0.5185	0.7615	0.5283

Table 49: The detailed versatility performance of Law task under the 33B model in different methods.

	Accuracy
ZeroShot	0.120
Full SFT	0.192
LoRA	0.128
Wise-FT	0.144
V-SoftMask	0.188
L1	0.140
L2	0.156
CoFiTune	0.152

Table 50: The detailed speciality performance of Math task under the 33B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6386	0.4684	0.5592	0.3953	0.4732	0.5744	0.7936	0.5553
Full SFT	0.5502	0.4313	0.5155	0.3745	0.4676	0.4707	0.7495	0.4990
LoRA	0.6205	0.4246	0.4992	0.3836	0.4618	0.3739	0.7772	0.5147
Wise-FT	0.6303	0.4572	0.5435	0.4004	0.4632	0.5288	0.7813	0.5436
V-SoftMask	0.5559	0.4341	0.5196	0.3801	0.4634	0.4782	0.7557	0.5031
L1	0.6140	0.4383	0.5198	0.3892	0.4688	0.4718	0.7496	0.5240
L2	0.5891	0.4348	0.5177	0.3836	0.4697	0.4615	0.7554	0.5138
CoFiTune	0.6264	0.4659	0.5490	0.3907	0.4638	0.5499	0.7916	0.5471

Table 51: The detailed versatility performance of Math task under the 33B model in different methods.

	BERTScore	Rouge	BLEU	Rouge-1	Rouge-2	Rouge-L	Spec.
ZeroShot	0.7095	0.4481	0.2832	0.4693	0.4239	0.4511	0.4802
Full SFT	0.7651	0.5395	0.3995	0.6042	0.4616	0.5528	0.5680
LoRA	0.7489	0.5078	0.3756	0.5773	0.4228	0.5233	0.5441
Wise-FT	0.7464	0.5044	0.3711	0.5735	0.4207	0.5192	0.5406
V-SoftMask	0.7648	0.5398	0.3988	0.6036	0.4628	0.5531	0.5678
L1	0.7435	0.5006	0.3672	0.5703	0.4151	0.5168	0.5371
L2	0.7523	0.5167	0.3853	0.5805	0.4339	0.5358	0.5514
CoFiTune	0.7502	0.5131	0.3784	0.5824	0.4293	0.5275	0.5472

Table 52: The detailed speciality performance of CAG task under the 33B model in different methods.

	Instruct.	Gen-Kn.	Gen-Rs.	Logiqa1	Logiqa2	OCNLI	Zh_Winograd	Vers.
ZeroShot	0.6386	0.4684	0.5592	0.3953	0.4732	0.5744	0.7936	0.5553
Full SFT	0.5235	0.4244	0.5064	0.3539	0.4389	0.4897	0.7434	0.4847
LoRA	0.6282	0.4248	0.5093	0.3801	0.4357	0.4483	0.7725	0.5207
Wise-FT	0.6169	0.4551	0.5391	0.3899	0.4582	0.5388	0.7695	0.5370
V-SoftMask	0.5288	0.4293	0.5122	0.3571	0.4426	0.4995	0.7498	0.4901
L1	0.6012	0.4328	0.5130	0.3632	0.4460	0.4939	0.7489	0.5156
L2	0.5746	0.4286	0.5095	0.3565	0.4441	0.4895	0.7479	0.5042
CoFiTune	0.6224	0.4647	0.5424	0.3915	0.4620	0.5245	0.7916	0.5432

Table 53: The detailed versatility performance of CAG task under the 33B model in different methods.

A.16 Module Importance for Speciality

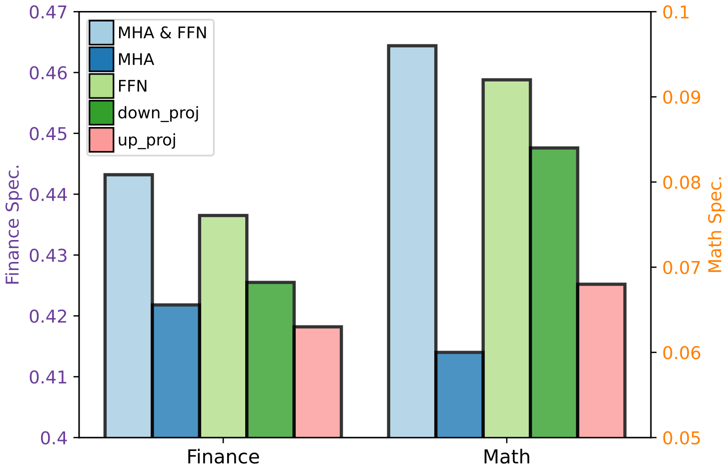

Figure 7: The Spec. score of different modules trained in all layers for Finance and Math tasks under the 7B model.

A.17 Exploring CF in LLM's Versatility

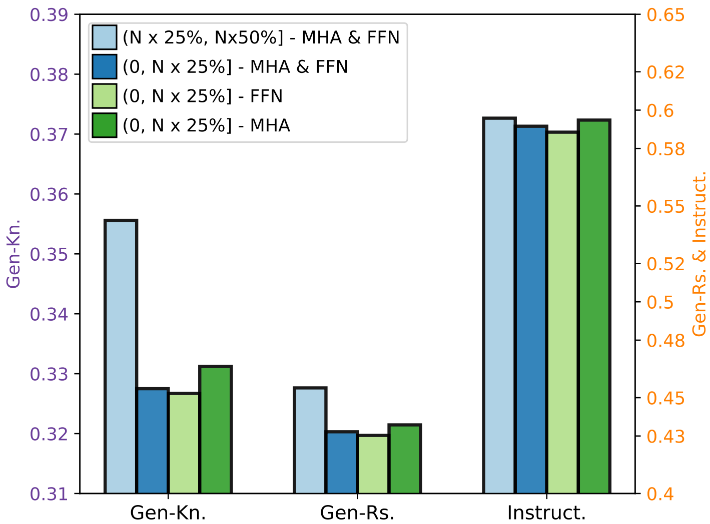

Figure 8: The Gen-Kn., Gen-Rs., and Instruct. scores in Math task under the 7B model.

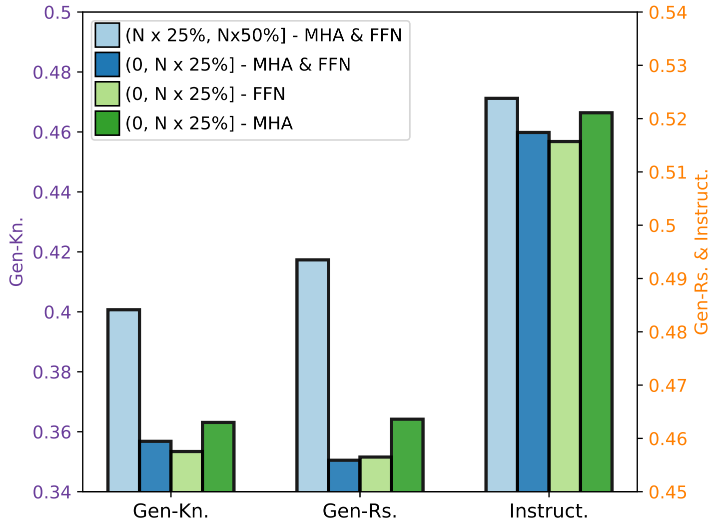

Figure 9: The Gen-Kn., Gen-Rs., and Instruct. scores in Finance task under the 13B model.

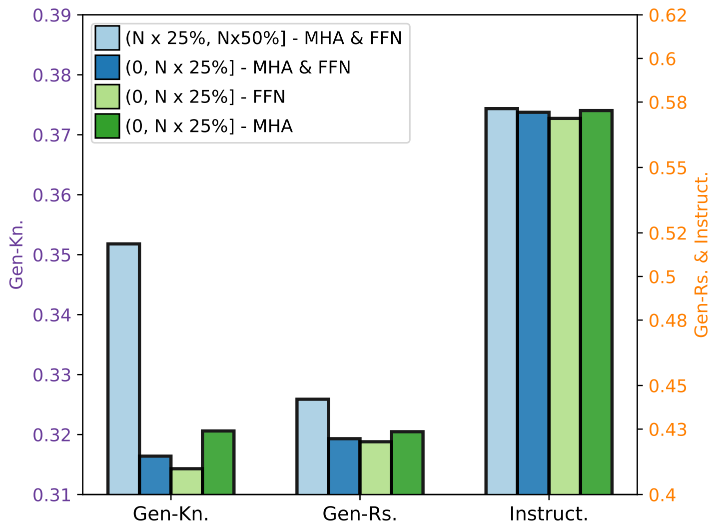

Figure 10: The Gen-Kn., Gen-Rs., and Instruct. scores in Finance task under the 7B model.

A.18 Details of Our Evaluation Setting

A.18.1 A New Strategy for Improving the Reliability of Evaluation Results

	Test set			Sample-Then-Paraphrase (Ours)		
	BERTScore	Rouge	BLEU	BERTScore	Rouge	BLEU
Full SFT	0.6325	0.2218	0.0992	0.7376	0.3973	0.2163
LoRA	0.6314	0.2239	0.1026	0.7159	0.3487	0.1527
Wise-FT	0.6293	0.2195	0.1003	0.7208	0.3597	0.1764
V-SoftMask	0.6318	0.2207	0.0984	0.7372	0.3980	0.2175
L1	0.6298	0.2202	0.0977	0.7181	0.3551	0.1737
L2	0.6306	0.2225	0.1015	0.7254	0.3649	0.1783
CoFiTune	0.6315	0.2206	0.0988	0.7222	0.3645	0.1788

Table 54: Comparison between directly using the test set and our sample-then-paraphrase strategy. We present the result using the 7B model in the finance task.

The exploration algorithm mentioned in Sec. 3.3 requires extensive experiments conducted to explore the distinct role of each layer range and modules within it. This necessitates a fast, cost-effective, and accurate automatic evaluation strategy. For Finance, Law, and CAG, we employ automatic generation metrics encompassing both semantic alignment and n-gram matching, including BERTScore (Zhang et al., 2020), Rouge (Lin, 2004), and BLEU (Papineni et al., 2002). The automatic results for Finance under the 7B model in the unseen test set are presented in Table 54. As depicted in Table 54, our observations indicate that the evaluation results across different methods for the unseen test set are similar and generally demonstrate suboptimal performance. This finding is not unique, as previous studies (Ovadia et al., 2023; Wang et al., 2023a) in question-answering have encountered similar issues[26], and diverse retrieval-augmented generation (RAG) approaches (Ovadia et al., 2023; Gao et al., 2023) are proposed to enhance LLMs' performance in addressing this issue. Another possible reason is the inadequacy of current automatic evaluation metrics in gauging LLMs' output for unseen and dynamic questions (Chang et al., 2023; Wang et al., 2023a).

Consequently, to enhance the reliability of the automatic evaluation results, we advocate for a novel evaluation approach when evaluating Finance, Law, and CAG[27] tasks: initially sampling instructions from the training set and subsequently paraphrasing them to form the test questions, i.e., sample-then-paraphrase strategy. Intuitively, this strategy facilitates valid automatic evaluation of LLMs' generation by testing in-domain knowledge while addressing concerns related to data contamination through test instruction rephrasing (Schaeffer, 2023; Oren et al., 2023).

A.18.2 Validation of the Automatic Results

To further validate the effectiveness of the automatic metrics used in assessing the quality of LLMs' generation, we randomly sample a set of outputs from the financial *CoFiTune* 7B model and evaluate their Spec. score with both GPT-4 and human annotators.

For human evaluation, we randomly select 100 samples and ask a qualified annotator[28] to assess each output's quality based on aspects such as accuracy, relevance, and coherence. Each score can range from 1 (very poor) to 5 (very good). For GPT-4 evaluation, we adhere to the same criteria and create a prompt instructing GPT-4 to assign a rating to each model output. An example of our evaluation prompt for GPT-4 is presented in Table 55.

Upon acquiring individual scores from both the annotator and GPT-4, we normalize each score to a range between 0 and 1. Subsequently, we employ the Inter Annotator Agreement (IAA) metric to assess

[26]Fine-tuned models become static data snapshots during training and may swiftly become inadequate for effectively supporting dynamic scenarios, making them less capable of handling unseen domain-specific questions.

[27]The test set of CAG task comprises both positive and negative samples, and their evaluation methods slightly differ; for more details, refer to the CAG task description in Appendix A.6.2.

[28]The annotator has scored above 600 on the College English Test 6 level (CET-6) and is compensated with 5 RMB per sample.

Prompt	You serve as an impartial evaluator. Please adhere to the criteria outlined below and assess the provided output on a scale ranging from 1 (very poor) to 5 (very good). Evaluate based on the following dimensions: Accuracy: Assess the truthfulness and factual correctness of the candidate's response. Relevance: Examine how well the response aligns with the topic of the question. Coherence: Evaluate how seamlessly the response integrates into the context, considering consistency with previous statements and overall flow of the answer. Please apply these criteria to the following question and output: Question: {Question} Output: {Output} Accuracy: Relevance: Coherence: Overall:

Table 55: The prompt we adopt for GPT-4 evaluation. We use "{Question}" and "{Output}" to replace the specific question and output.

the concordance between our automatic Spec. scores and the scores obtained from both the annotator and GPT-4. As illustrated in Table 56, our Spec. score reveals a substantial level of agreement with evaluations conducted by both human annotators and GPT-4. This alignment underscores the efficacy and robustness of the Spec. score as a reliable metric for assessing the quality of outputs generated by LLMs in our evaluation setting (mentioned in Appendix A.18.1).

	GPT-4	Human Annotators
k	0.689	0.653
p_0	0.800	0.756

Table 56: Inter-Annotator Agreement (IAA) measured by Cohen's Kappa, and the agreement rate between GPT-4 score, human annotators and our Spec. score.