CUSTOMIZABLE COMBINATION OF PARAMETER-EFFICIENT MODULES FOR MULTI-TASK LEARNING

Haowen Wang, Tao Sun, Congyun Jin, Yingbo Wang, Yibo Fan, Yunqi Xu, Yuliang Du & Cong Fan*
Ant Group
Shanghai, China
{wanghaowen.whw,suntao.sun,jincongyun.jcy,wangyingbo.wyb,fanyibo.fyb,
xuyunqi.xyq,duyuliang.dyl,fancong.fan}@antgroup.com

ABSTRACT

Modular and composable transfer learning is an emerging direction in the field of Parameter Efficient Fine-Tuning, as it enables neural networks to better organize various aspects of knowledge, leading to improved cross-task generalization. In this paper, we introduce a novel approach Customized Tuning (C-Tuning) that combines task-common skills and task-specific skills, while the skill parameters being highly parameterized using low-rank techniques. Each task is associated with a customizable number of exclusive specialized skills and also benefits from skills shared with peer tasks. A skill assignment matrix is jointly learned. To evaluate our approach, we conducted extensive experiments on the Super-NaturalInstructions and the SuperGLUE benchmarks. Our findings demonstrate that C-Tuning outperforms fully-shared, task-specific, and skill-indistinguishable baselines, significantly enhancing the sample efficiency in multi-task learning scenarios.

1 INTRODUCTION

As the number of parameters in Large Language Models (LLMs) continues to grow, training these models efficiently with limited computational resources has become a challenge. In recent years, there has been a shift towards employing Parameter Effective Fine-Tuning (PEFT) methods to address this issue. Examples of such methods include LoRA (Hu et al., 2022), AdaLoRA (Zhang et al., 2023a), and $(IA)^3$ (Liu et al., 2022). These methods focus on fine-tuning the adapter while freezing the pre-trained model, effectively reducing the computational cost. By selectively updating only a portion of the model parameters, PEFT methods enable efficient training and utilization of large foundation models. This line of approaches allows for more effective use of resources while maintaining the performance of the pre-trained model on downstream tasks (Hu et al., 2022). However, despite the popularity and widely adoption of PEFT methods, the learning effectiveness of such methods, especially in multi-task scenarios, is under explored.

LLMs are famous for their extraordinary capabilities on solving multiple tasks in zero-shot or few-shot manners (Brown et al., 2020). Basic PEFT methods mentioned earlier don't take the multitask essence of real-world data into account and rely heavily on the base foundation model's capacities on the multitask generalization. Building upon the basic PEFT methods, various training approaches designed for Multi-Task Learning (MTL) have been proposed (Pfeiffer et al., 2020; Vu et al., 2021; Asai et al., 2022; Chronopoulou et al., 2023; Zadouri et al., 2023). One simple solution is to perform multitask training by training a large model on a combination of multiple tasks. This involves training the model on the union of training tasks and subsequently evaluating its performance on different testing tasks (Ye et al., 2021; Liu et al., 2022). However, this approach overlooks the relationships between the tasks and is vulnerable to negative transfer, where the gradients associated with different tasks are misaligned (Wang et al., 2020). This misalignment of gradients can lead to sub-optimal performance and hinder the effective utilization of learned knowledge across tasks.

*Corresponding Author

To enhance sample efficiency, MoLoRA (Zadouri et al., 2023) has been introduced. MoLoRA successfully applies the Mixture-of-Expert (MoE) architecture to PEFT methods and improves the model's generalization capacity across various tasks by jointly learning multiple LoRA instances. MoLoRA views each LoRA as a lightweight expert, following MoE framework, and thus allows for more specialized adaptation to different tasks from shared knowledge learnt through parallel instances.

In a recent work, Ponti et al. (2022) developed Polytropon (`Poly`) to tackle the challenges associated with multitask learning. The central idea of `Poly` is to consider each task-specific adapter as a composition of reusable skillset of basic adapters or modules. Specifically, `Poly` jointly learns an inventory of adapters (for example, LoRA) and a simple routing vector that selects and combines a (variable-size) subset of adapters for each individual task. This approach significantly improves the efficiency of sample sharing and utilization between multiple tasks. Although not mentioned explicitly in the original work (Ponti et al., 2022), we argue that each adapter in `Poly` can be viewed as a lightweight expert and then the whole structure follows the same pattern as Multi-gate Mixture-of-Experts (MMoE) (Ma et al., 2018), a well-known MTL framework. To be noted, in Ponti et al. (2022) also introduced a structure MoE-LoRA when treating all tasks the same, which is a simplified version of MoLoRA where the routing is controlled by a function of hidden states instead.

Based on `Poly`, a subsequent research titled Multi-Head Routing (`MHR`) (Caccia et al., 2022) was proposed. `MHR` extends the concept of parallel thinking to low-rank decomposition and introduces further improvements to the basic unit of the adapter. By leveraging low-rank decomposition, `MHR` enhances the model's ability to generalize across different tasks. The parallel improvements in the adapter's basic units allow for more efficient adaptation to different tasks while still benefiting from shared knowledge.

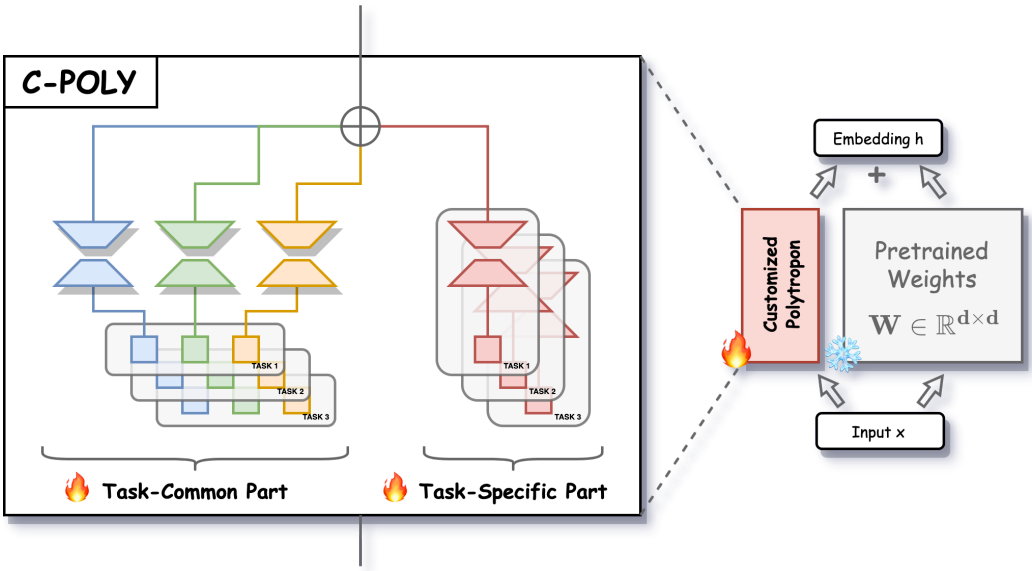

Figure 1: Overview of Customized Tuning (`C-Tuning`) framework

Inspired by Customized Sharing Control (CGC) and Progressive Layered Extraction (PLE) (Tang et al., 2020), our research has made additional improvements over `Poly`. We assume that in MTL, many tasks do share transferable knowledge, while each of them requires discriminative abilities. Based on this assumption, we explicitly divide modular skills into task-common skill modules and task-specific skill modules. We propose **Customized Tuning** (**`C-Tuning`**), where for each task, two components are learnt jointly, a task-specific skill module and a task-common skill module. This allows each task to be characterized by not only a shared task-common module as in `Poly` and `MHR`, but also a unique subset of skills, mitigating the effect of negative transferring and leading to improved multi-task performance. Furthermore, `C-Tuning` promotes interpretability by learning an explicit hierarchy of tasks based on the skills they select, which provides insights into the rela-

tionship between tasks and the skills needed to perform them effectively. Additionally, we observe the task-specific instantiation of a neural network by combining specific parameters of the active skills. This reduces memory usage and facilitates more effective training, as only the necessary parameters associated with active skills are utilized. Overall, our approach enhances the efficiency and effectiveness of modular skill multi-task learning, enabling better performance, interpretability, and resource utilization.

In Section 3, we conducted extensive experiments on multiple datasets and model architectures. The results of these experiments have consistently demonstrated our method `C-Tuning` has surpassed the performance of the most advanced PEFT methods and achieved state-of-the-art (SOTA) results in different multitasking scenarios.

2 METHODOLOGY

The proposed unified MTL framework `C-Tuning` is shown in Figure 1, which aims to enhance sample efficiency for each task by leveraging strengths from all other tasks while keeping task-specific abilities. Suppose there are T tasks and for each task, task-specific data input x^t, $t \in \{1, 2, \ldots, T\}$. The MoE-like structure consists of adapter modules (or experts), $\mathbf{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_{|\mathbf{\Phi}|}\}$ and each adapter ϕ_i can be regarded as a function of the input data x^t. The major improvement of `C-Tuning` is to explicitly categorize adapter modules $\mathbf{\Phi}$ into two separate parts:

- A task-common skillset: $\mathbf{\Phi}_A = \{\phi_1, \phi_2, \ldots, \phi_A\}$ having A adapters.
- A task-specific skillset: $\mathbf{\Phi}_B^t = \{\phi_1^t, \phi_2^t, \ldots, \phi_B^t\}$ having B adapters for each task t.

In total, there are $|\mathbf{\Phi}_A| + T \times |\mathbf{\Phi}_B^t| = A + T \times B$ adapters. For a simplified yet generalizable discussion, we would set $B = 1$ to keep only one task-specific adapter in the following experiments.

The combined output of the `C-Tuning` adapter modules for each task input x^t can be expressed in Equation 1 with w_i representing the learnable weight of each adapter's output.

$$\underbrace{\sum_{i=1}^{A} w_i^t \phi_i(x^t)}_{\text{Task-Common}} + \underbrace{\sum_{j=1}^{B} w_j^t \phi_j^t(x^t)}_{\text{Task-Specific}} = \sum_{i=1}^{A} w_i^t \phi_i(x^t) + w^t \phi^t(x^t) \tag{1}$$

In the task-common part, the set of adapters ϕ_i are shared across all tasks, while the weights w_i^t are exclusive to each task t. In contrast, both the weights w_i^t and adapters ϕ_i^t are customized for each individual task t in the task-specific part.

Following the notation above, various MoE-like PEFT structures can be mathematically formulated together in Table 1. Both the MoE and MMoE models only consist of the task-common part of Equation 1: in the conventional MoE approach, tasks are not differentiated, leading to shared parameters across all tasks; the MMoE framework assigns task-specific weights for each individual task, while still maintaining a shared pool of experts or adapters.

In `C-Tuning`, the weights associated with each adapter can be represented together as one allocation matrix $\boldsymbol{W} \in \mathbb{R}^{T \times (A+T)}$ when $B = 1$. This matrix can be further decomposed into two distinct components: $\boldsymbol{W}_A \in \mathbb{R}^{T \times A}$ and $\boldsymbol{W}_B \in \mathbb{R}^{T \times T}$:

$$\boldsymbol{W} = \left[\begin{array}{c|c} \boldsymbol{W}_A & \boldsymbol{W}_B \end{array}\right] \tag{2}$$

$$= \left[\begin{array}{cccc|cccc} w_1^1 & w_2^1 & \cdots & w_A^1 & w^1 & 0 & 0 & 0 \\ w_1^2 & w_2^2 & \cdots & w_A^2 & 0 & w^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ w_1^T & w_2^T & \cdots & w_A^T & 0 & 0 & \cdots & w^T \end{array}\right] \tag{3}$$

To optimize the learning process in skill acquisition, we have employed different learning methods for each component of the allocation matrix. Additionally, we have incorporated low-rank approximations to further enhance the parameter efficiency.

Table 1: Comparison between different MoE-like PEFT methods

MoE Structures	PEFT Methods	Task Output
Conventional MoE	MoLoRA (Zadouri et al., 2023), MoE-LoRA (Ponti et al., 2022)	$\sum_{i=1}^{A} w_i \phi_i(x^t)$
MMoE (Ma et al., 2018)	`Poly` (Ponti et al., 2022), `MHR` (Caccia et al., 2022)	$\sum_{i=1}^{A} w_i^t \phi_i(x^t)$
CGC (Tang et al., 2020), PLE (Tang et al., 2020)	**Our Method** `C-Tuning`	$\sum_{i=1}^{A} w_i^t \phi_i(x^t) + w^t \phi^t(x^t)$

2.1 Task-Common Skills Learning

Task-common skills are universally applicable skills that all tasks can leverage. Previous research has focused on identifying the effectiveness of general skills modules for specific tasks by employing modular concepts at the structured input level inspired by cognitive mechanisms (Bengio, 2017; Ponti et al., 2022; Caccia et al., 2022; Zadouri et al., 2023). This concept has been translated into Softmax for cross-module and top-k selection in practical implementation.

Therefore, following Ponti et al. (2022) and Caccia et al. (2022), we utilize a task-common allocation matrix $\boldsymbol{W}_A \in \{0,1\}^{T \times A}$ with a uniform initialization. This matrix is employed to achieve the soft partitioning of general skills. Each element w_i^t in $\boldsymbol{W}_A$ is a binary value that indicates whether a particular task t activates the adapter module ϕ_i. However, since discrete binary matrices like $\boldsymbol{W}_A$ are non-differentiable, learning cannot be accomplished through gradient descent. To overcome this limitation, we adopt the Gumbel-sigmoid approach (Maddison et al., 2016; Jang et al., 2016), which allows us to obtain a set of continuously relaxed Bernoulli distributions. This approach guarantees both randomness and the ability to perform differentiable sampling:

$$\hat{w_i^t} = \sigma\left[\log \frac{\sigma(w_i^t)\mathrm{u}}{(1-\sigma(w_i^t))(1-\mathrm{u})}\right], \quad \mathrm{u} \sim \mathcal{U}(0,1) \tag{4}$$

2.2 Task-Specific Skills Learning

Specialized skills refer to modular skills that acquire the distinctive attributes of each task. In complex and interconnected multitasking scenarios, a seesaw phenomenon commonly arises. In multitasking learning mode, there is often a trade-off between enhancing specific tasks' effectiveness and compromising others' effectiveness (Wang et al., 2020). To address this, we explicitly differentiate between shared and exclusive skill modules. This separation allows us to amplify the inherent characteristics of individual tasks.

We initialize the task-specific skill allocation $\boldsymbol{W}_B \in \mathbb{R}^{T \times T}$ as a unit diagonal matrix. Notably, during the actual training process, although only the diagonal of $\boldsymbol{W}_B$ are weights we concerned, entries off the diagonal are also subject to potential updates. These side-effects indicate that while solving the current task, the exclusive specialized skills of other tasks can be leveraged without influencing the parameter values of specialized skills for those tasks.

2.3 Parameter Efficiency

We accomplished an efficient parameterization of skill modules by employing low-rank techniques. Every adapter in our `C-Tuning` experiments is Low-Rank Adapter (LoRA) (Hu et al., 2022). LoRA is a straightforward yet effective structure specifically tailored for Transformer-based models (Vaswani et al., 2017). The idea behind LoRA is relatively straightforward. LoRA decomposes each weight matrix of the linear transformation in Transformers into the multiplication of two low-

rank matrices. In other words, the linear projection $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ can be represented as follows, disregarding the bias part:

$$\boldsymbol{h}_{l+1} = \boldsymbol{h}_l \left[\boldsymbol{W}_l + \Delta\boldsymbol{W}\right] = \boldsymbol{h}_l \left[\boldsymbol{W}_l + \boldsymbol{W}_{down}\boldsymbol{W}_{up}\right] \tag{5}$$

Instead of $\Delta\boldsymbol{W} \in \mathbb{R}^{d\times d}$, two much smaller matrices $\boldsymbol{W}_{down} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{W}_{up} \in \mathbb{R}^{r\times d}$, with $r \ll d$, are obtained through gradient descend optimization. Through the adoption of LoRA, updating each linear layer in the model only requires $2 \times r \times d$ parameters in the calculation, as opposed to the original $d \times d$ parameters. This results in a notable enhancement in parameter efficiency, enabling faster training even with limited computing resources.

As examined in Hu et al. (2022), LoRA can be applied to various components of Transformers, such as query, key, value, and feed-forward layers, while the choice of rank r does not hold significant importance. This suggests that LoRA exhibits versatility in its applicability. In our experiments, we replaced all query, key, and value layers as `C-Tuning`, a combination of multiple LoRAs.

3 Empirical Experiments

3.1 Experimental Setup

In order to evaluate the efficacy of our proposed unified MTL framework `C-Tuning`, which incorporates both task-common and task-specific skills, we conducted experiments on two publicly available multitasking benchmarks: SuperGLUE (Wang et al., 2019) and Super Natural-Instructions (Super NI) (Wang et al., 2022). The SuperGLUE benchmark is a widely acceptable benchmark for evaluating general-purpose language understanding. In our experiments, seven distinct tasks were selected from the benchmark that can be effectively evaluated using the accuracy metric. The Super NI dataset, as a meta-dataset (Triantafillou et al., 2019), covers a wide range of 76 distinct task types within the field of natural language processing and comprises over 1,600 diverse NLP tasks. During the experiments, 100 tasks were randomly selected, and for each task, 1000 samples were randomly selected for training and another 100 were selected for evaluation purpose. To ensure comparability, our sampling method follows the identical approach as described in Ponti et al. (2022). To evaluate the effectiveness of the trained model, we employed various metrics for all selected tasks, including Exact Match (EM) and Rouge metrics (Lin, 2004), including Rouge-1, Rouge-L, and Rouge-LSum.

To verify the universal effectiveness of our multitasking learning approach `C-Tuning`, we chose T5 Version 1.1 - LM Adapted (T5) (Raffel et al., 2020), FLAN-T5 (Chung et al., 2022) and GLM (Du et al., 2021) as the base models.

In our study, we thoroughly compared our proposed approach, `C-Tuning`, and several existing MoE-like PEFT methods. The methods we compared against include LoRA, MoE-LoRA, `Poly` and `MHR`. The comparison allowed us to analyze and evaluate the performance and effectiveness of our framework relative to these established PEFT methods.

3.2 Training Details

In our experiments, we applied PEFT methods to all query, key, value matrices within every attention layer in the base models. In the case of vanilla LoRA, we set the rank of the low-rank approximation, $r = 8$. For all MoE-like tuning methods, we utilized in total 4 parallel LoRAs (experts) with $r = 2$. In `C-Tuning`, we set $A = 3$ LoRA for task-common skills and $B = 1$ LoRA for task-specific skills. This decision was made to ensure a comparable number of training parameters across all methods.

We trained our model for 1 epoch only with a batch size of 4 on both Super NI and SuperGLUE datasets during training. The AdamW optimizer (Loshchilov & Hutter, 2017) was used, with a learning rate of $5e^{-5}$. We also employed the linear decay strategy (Loshchilov & Hutter, 2016) as the learning rate scheduler with a weight decay of 0.01 and a warmup ratio of 0.06. All experiments were conducted on a single NVIDIA Tesla A100 graphics card.

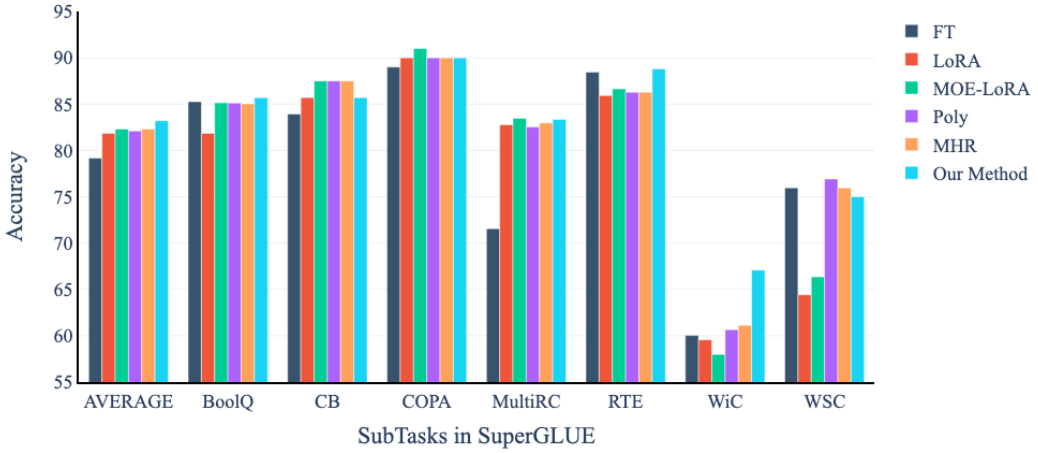

Figure 2: FLAN-T5-Large with different PEFT methods on SuperGLUE benchmark, compared with Full Fine-tuning (FT). We reported overall averaged (AVERAGE) and task-specific accuracy for all sub-tasks.

3.3 Main Results and Discussion

3.3.1 Analysis of Balanced Multitask Learning

In Figure 2, we present a comparative analysis of various fine-tuning methods across multiple tasks within the SuperGLUE benchmark. When tuning with full parameters (FT), the overall average accuracy is the lowest among all approaches because of relatively poor performance in the MultiRC sub-task. This phenomenon, known as the seesaw effect, a manifestation of the negative transfer problem when tackling multiple tasks concurrently, holds significant importance in the domain of multi-task learning (Pan & Yang, 2010; Sun et al., 2017; Tang et al., 2020). Our method `C-Tuning`, on the other hand, demonstrates constant improvement over all sub-tasks thanks to the task-specific skill learning module. The results reveal that `C-Tuning` can effectively mitigate the negative transfer and seesaw effect issues. As shown in Appendix A.4, we conducted experiments on FLAN-T5-XL (2B) and found that the base model with larger parameter values has a stronger fitting ability for multiple tasks. The experimental results showed that our method can bring stable improvements in reducing negative migration. Consequently, our approach exhibits substantial superiority over other methods.

3.3.2 Evaluation on More Models and More Tasks

The effectiveness of different PEFT methods are evaluated on two architectures: T5 (Encoder-Decoder) and GLM (Decoder-Only). Table 2 presents the performance of these methods on a dataset consisting of 7 tasks from the SuperGLUE. We display the evaluation results for the 7 sub-tasks individually, as well as their average performance. Table 3 compares the performance of different PEFT methods on the SuperNI. The indicators in the table represent the average performance evaluation over 100 tasks.

The results are two-folded. Firstly, it highlights that MoE-LoRA consistently demonstrates improvement over LoRA, attributed to the enhanced parameter flexibility from the MoE structure. We compared FLAN-T5-Large, T5-Large, and GLM-10B, and the results showed that it can significantly alleviate the phenomenon of negative migration in both encoder-decoder and decoder-only architectures and improve the flexibility of adaptive parameters compared to `LoRA` and `Poly`. In the comparison of FLAN-T5-Large, although it has undergone a large number of pre-training with similar instruction samples, our method can still bring some improvement. The ablation experiments

Table 2: FLAN-T5-Large and GLM-10B with different adaptation methods on the SuperGLUE dataset. We report the overall (matched and mismatched) accuracy for BoolQ, CB, COPA, MultiRC, RTE, WiC and WSC. Higher is better for all metrics.

Base Model	PEFT Method	AVG	BoolQ	CB	COPA	MultiRC	RTE	WiC	WSC
FLAN-T5-Large	LoRA	81.85	81.85	85.71	90.00	82.78	85.92	59.56	64.42
	MOE-LoRA	82.31	85.14	**87.50**	**91.00**	**83.46**	86.64	57.99	66.35
	`Poly`	82.09	85.11	**87.50**	90.00	82.53	86.28	60.66	**76.92**
	`MHR`	82.31	85.02	**87.50**	90.00	82.96	86.28	61.13	75.96
	Our Method	**83.21**	**85.69**	85.71	90.00	83.35	**88.81**	**67.08**	75.00
GLM-10B	LoRA	52.05	60.98	46.38	65.70	62.43	57.37	39.15	32.32
	MoE-LoRA	53.86	63.31	45.02	63.41	64.01	61.22	40.35	39.66
	`Poly`	56.99	64.65	52.17	65.54	65.66	62.15	41.71	47.08
	`MHR`	56.92	64.85	50.79	66.36	65.79	62.75	42.35	45.58
	Our Method	**62.26**	**67.31**	**60.38**	**70.04**	**67.90**	**68.01**	**48.71**	**53.42**

Table 3: FLAN-T5-Large, T5-Large and GLM-10B with different adaptation methods on the 100 randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics.

Base Model	PEFT Method	rouge1	rougeL	rougeLsum
FLAN-T5-Large	LoRA	68.26	67.42	67.42
	MoE-LoRA	68.59	67.76	67.75
	Poly	68.45	67.60	67.58
	MHR	68.84	67.77	67.78
	Our Method	**68.69**	**67.80**	**67.82**
T5-Large	LoRA	34.16	33.64	33.65
	MoE-LoRA	36.82	36.13	36.15
	Poly	43.04	42.05	42.09
	MHR	44.24	43.32	43.34
	Our Method	**49.34**	**48.50**	**48.51**
GLM-10B	LoRA	43.16	42.04	42.09
	MoE-LoRA	45.97	44.79	44.89
	Poly	47.96	46.80	46.80
	MHR	48.53	47.34	47.33
	Our Method	**49.53**	**48.45**	**48.45**

in Appendix A.2 on the T5 and FLAN-T5 series model of different scales (Large, XL, XXL) showed that our methods remained robust and significant as model parameters increased.

Secondly, the results demonstrate that our method `C-Tuning` has achieved optimal performance in both architectures and on the SuperGLUE and SuperNI datasets. The ablation experiments in Appendix A.1 on the number of tasks (10-50-100) conducted on GLM-10B and FLAN-T5-Large indicate that our method is still significantly effective when the number of tasks increases. The explicit separation of task-specific skills and task-common skills in our design enables the skill modules to capture task-specific differences while sharing abstracted general skill modules effectively. Due to the explicit separation, the negative transfer phenomenon has been significantly reduced as in Tang et al. (2020), which can be verified on both datasets and becomes more pronounced as the number of learning tasks increases.

3.3.3 PARAMETER EFFICIENCY ANALYSIS

Figure 3 exhibits intriguing observations pertaining to various PEFT techniques within a constrained temporal epoch. We assessed PEFT methodologies across varying parameter magnitudes, gauging their efficacy on the SuperGLUE multitask dataset with FLAN-T5-Large as the foundational model. Furthermore, we juxtaposed these outcomes against those derived from full parameter fine-tuning. It is evident that our approach attains superior performance among all PEFT methodologies possessing identical parameter scales and training epochs. Notably, `C-Tuning` even outperforms other methods with more parameters. Our methods explicitly segregate shared and proprietary skills and effectively ensure parameter efficiency in multi-task learning under equivalent training settings. This trend is also observed in the SuperNI benchmark evaluation. As a result, we assert that adopting both task-common skills learning and task-specific skills learning as a paradigm constitutes a robust strategy for achieving parameter efficiency in adapting multiple tasks. Furthermore, our approach surpasses the performance of full parameter fine-tuning while utilizing fewer parameters, exclusively updating routing parameters and adapters.

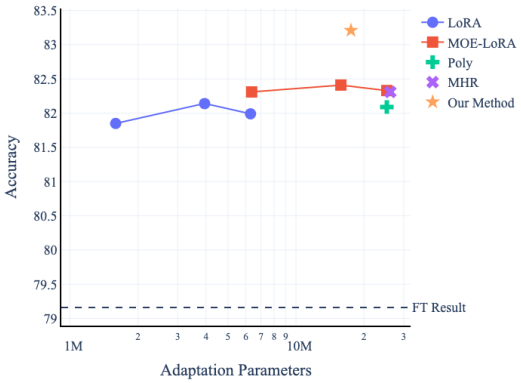

Figure 3: Accuracy of PEFT methods and Full Fine-tuning on SuperGLUE dataset when applied on T5-Large. The X-axis shows the trainable parameter count during the fine-tuning process.

3.3.4 DEEPER INSIGHTS INTO C-TUNING

As outlined in Section 3.3.2, our dual-skill framework, `C-Tuning`, consistently delivers robust enhancements across various architectures, task scales, and model sizes. Here, we try to explore the intrinsic attributes of `C-Tuning` that drive these performance gains.

Explicit skill separation enhances knowledge sharing Task-specific skills enable the task-common component to focus on task similarities, while they manage the distinct elements of each task. Figures 5, 6, and 7 demonstrate the task-common skill allocation W_A in certain layers of the `C-Tuning` GLM-10B model for SuperNI-10, SuperNI-50, and SuperNI-100 respectively. After proper normalization, these matrices reveal clear differences in skill allocation for distinct tasks. Additionally, we conducted task clustering based on the skill allocations learned in all layers of the GLM-10B model trained on SuperNI-100, with and without the task-specific component (i.e., comparing `C-Tuning` and `Poly`). The clustering outcomes are shown as dendrograms in Figures8 and 9, which suggest the enhanced performance of `C-Tuning` is due to a more balanced task hierarchy, facilitating more effective knowledge transfer among similar tasks and improved distinction of unrelated ones.

Equilibrium in skill allocation is crucial The introduction of `C-Tuning` prompts the need to balance task-common and task-specific parameter allocation.We must establish the optimal A and B values. In Appendix A.3, we explore various (A, B) pairings, maintaining a fixed total parameter count and confirm that task-specific skills significantly boost model performance. However, allocating enough parameters to task-common skills is crucial for shared knowledge acquisition.

Excessive focus on task-specific parameters can impede the learning process, potentially causing overfitting and hindering the model's ability to recognize similarities across tasks, which may lead to suboptimal performance. A balanced parameter distribution promotes clear task distinction and helps prevent overfitting, preserving the model's generalization capabilities. Identifying the ideal parameter ratio and configuration is a critical aspect for `C-Tuning` and may vary depending on the nature of the tasks and datasets involved.

4 Related Works

4.1 Parameter-Efficient Fine-Tuning

Numerous researchers have proposed incorporating adapters within neural networks, strategically placed between existing layers, and enforcing weight constraints on these adapters. LoRA (Hu et al., 2022), for instance, advocates for fine-tuning the model by learning low-rank matrix weights and aligning them with sovereignty. Building upon LoRA, $(IA)^3$ (Liu et al., 2022) offers further enhancements by introducing a relatively modest number of novel parameters. In the context of model adjustment, prefix tuning emerges as a technique that exclusively optimizes a small segment of continuous task-specific vectors, thereby strengthening downstream task optimization.

4.2 Modular Multi-task Learning

Modular neural network architectures (Jacobs et al., 1991a) offer benefits like positive transfer, compositionality, and parameter efficiency (Zhang et al., 2023b). They consist of modules (which represent skills that can be combined and updated independently), a routing function (to select modules per task or example with variants like fixed, learned hard, or learned soft routing (Rosenbaum et al., 2017; Jacobs et al., 1991b; Fernando et al., 2017)), and an aggregation function (to merge outputs of active modules, often as a learnable network). Notably, within the realm of NLP, Fedus et al. (2021)successfully extended the pre-training of large language models to trillions of parameters by leveraging the MoE architecture. Previous studies (Rajendran et al., 2015; Ponti et al., 2022; Kingetsu et al., 2021; Kudugunta et al., 2021) have explored approaches to enforce parameter reuse and modularity in multitasking learning. Rajendran et al. (2015) trained individual modules for each task and subsequently learned how to reuse these modules.

4.3 Language Models

The Transformer architecture serves as a fundamental framework for sequence pair modeling. Building upon this foundation, Radford & Narasimhan (2018) employed a stack of Transformers to effectively model autoregressive languages through the deployment of encoders and decoders. BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) are classic text modeling methodologies, both relying on Transformer units pre-trained on vast amounts of textual data. The encoder-only architecture model is particularly suited for comprehension-based tasks, whereas generative tasks benefit from both encoder-decoder and decoder-only architecture models due to their autoregressive nature (Fu et al., 2023; Sarrouti et al., 2022).

5 Conclusion

In this article, we introduce a novel paradigm for PEFT called Customized Tuning `C-Tuning`. By explicitly distinguishing between task-common and task-specific skills, our method enables efficient multi-task fine-tuning on large language models, even with limited computational resources. Our approach addresses the challenge of resource limitations and allows for efficient training. The separation of exclusive and general skills effectively mitigates the seesaw problem and negative transfer commonly encountered in multitasking learning, leading to superior overall performance, which also offers compelling interpretability. Extensive experimental evaluations demonstrate the effectiveness of our proposed method, surpassing existing PEFT baselines and achieving state-of-the-art performance. These results highlight the potential and significance of our unified multi-task learning framework `C-Tuning` in the field of parameter-efficient multi-task learning.

REFERENCES

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6655–6672, 2022.

Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL `https://arxiv.org/abs/2005.14165`.

Lucas Caccia, Edoardo Ponti, Lucas Liu, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. Multi-head adapter routing for data-efficient fine-tuning. *arXiv preprint arXiv:2211.03831*, 2022.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*, 2023.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

Robert A Jacobs, Michael I Jordan, and Andrew G Barto. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive science*, 15 (2):219–250, 1991a.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991b.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Hiroaki Kingetsu, Kenichi Kobayashi, and Taiji Suzuki. Neural network module decomposition and recomposition, 2021.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3577–3599, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.304. URL https://aclanthology.org/2021.findings-emnlp.304.

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

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pp. 1930–1939, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220007. URL https://doi.org/10.1145/3219819.3220007.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

Edoardo M Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining modular skills in multitask learning. *arXiv preprint arXiv:2202.13914*, 2022.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Janarthanan Rajendran, Aravind Srinivas, Mitesh M. Khapra, P Prasanna, and Balaraman Ravindran. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain, 2015.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning, 2017.

Mourad Sarrouti, Carson Tao, and Yoann Mamy Randriamihaja. Comparing encoder-only and encoder-decoder transformers for relation extraction from biomedical texts: An empirical study on ten benchmark datasets. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pp. 376–382, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bionlp-1.37. URL https://aclanthology.org/2022.bionlp-1.37.

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in bayesian neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2017.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pp. 269–278, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832. doi: 10.1145/3383313.3412236. URL https://doi.org/10.1145/3383313.3412236.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096, 2019. URL http://arxiv.org/abs/1903.03096.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*, 2021.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.340. URL https://aclanthology.org/2022.emnlp-main.340.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*, 2020.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. *arXiv preprint arXiv:2104.08835*, 2021.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning, 2023.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023a.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers, 2023b.

A ADDITIONAL RESULTS

A.1 PEFT PERFORMANCE ON DIFFERENT TASK NUMBERS

We compared and evaluated the performance of different PEFT multitasking tuning frameworks on different bases and with different task quantities. Table 4, Table 5 and Table 6 show the performance of multi task tuning with different task quantities based on T5-Large, Flan-T5-Large, and GLM-10B models respectively. It can be clearly seen that our method can achieve significant improvements under different numbers of tasks.

It is worth mentioning that FLAN-T5-Large is a model obtained from large-scale training of the T5-Large architecture model on FLAN data, and its training set may have some similarities or overlaps with the NI multitasking instruction data training set, which makes its performance very high and reduces the gap between different PEFT methods.

Table 4: Performance on T5-Large with different numbers of randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics

Tasks Numbers	PEFT Method	rouge1	rougeL	rougeLsum
10	LoRA	14.22	14.12	14.19
	MoE-LoRA	16.75	16.71	16.76
	Poly	17.34	17.31	17.38
	MHR	17.17	17.11	17.16
	Our Method	**42.62**	**42.47**	**42.60**
50	LoRA	32.58	31.64	31.59
	MoE-LoRA	39.50	38.47	38.46
	Poly	46.13	44.25	44.28
	MHR	47.27	45.43	45.39
	Our Method	**53.39**	**51.68**	**51.63**
100	LoRA	34.16	33.64	33.65
	MoE-LoRA	36.82	36.13	36.15
	Poly	43.04	42.05	42.09
	MHR	44.24	43.32	43.34
	Our Method	**49.34**	**48.50**	**48.51**

A.2 PEFT PERFORMANCES ON DIFFERENT SCALE OF BASE MODEL

Table 7 and Table 8 show the performance of T5 and FLAN-T5 models with different parameter quantities under different PEFT methods, respectively. It can be seen that our method has a robust improvement on models with different parameter quantities, and as the parameter quantity increases, the magnitude of the improvement gradually increases.

Table 5: Performance on FLAN-T5-Large with different numbers of randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics

Tasks Numbers	PEFT Method	rouge1	rougeL	rougeLsum
10	LoRA	67.82	67.01	67.03
	MoE-LoRA	67.95	67.12	67.15
	Poly	68.10	67.29	67.33
	MHR	77.49	77.25	77.36
	Our Method	**77.73**	**77.58**	**77.61**
50	LoRA	70.66	69.10	69.03
	MoE-LoRA	70.81	69.25	69.21
	Poly	70.76	69.23	69.15
	MHR	70.92	69.39	69.33
	Our Method	**71.17**	**69.68**	**69.62**
100	LoRA	68.26	67.42	67.42
	MoE-LoRA	68.59	67.76	67.75
	Poly	68.45	67.60	67.58
	MHR	68.84	67.77	67.78
	Our Method	**68.69**	**67.80**	**67.82**

Table 6: Performance on GLM-10B with different numbers of randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics

Tasks Numbers	PEFT Method	rouge1	rougeL	rougeLsum
10	LoRA	30.64	30.40	30.42
	MoE-LoRA	33.92	33.79	33.77
	Poly	34.53	34.41	34.31
	MHR	33.63	33.47	33.47
	Our Method	**43.74**	**43.72**	**43.65**
50	LoRA	34.16	33.00	32.98
	MoE-LoRA	39.87	38.63	38.55
	Poly	44.81	43.09	43.07
	MHR	45.32	43.62	43.56
	Our Method	**53.17**	**51.27**	**51.32**
100	LoRA	43.16	42.04	42.09
	MoE-LoRA	45.97	44.79	44.89
	Poly	47.96	46.80	46.80
	MHR	48.53	47.34	47.33
	Our Method	**49.53**	**48.45**	**48.45**

Table 7: T5 with different parameter scales' performances with different adaptation methods on the 100 randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics

Base Model	**PEFT Method**	**Rouge1**	**RougeL**	**RougeLsum**
T5-Large	LoRA	34.16	33.64	33.65
	MoE-LoRA	36.82	36.13	36.15
	Poly	43.04	42.05	42.09
	MHR	44.24	43.32	43.34
	Our Method	**49.34**	**48.50**	**48.51**
T5-XL	LoRA	34.93	34.34	34.40
	MoE-LoRA	39.78	38.83	38.87
	Poly	43.61	42.61	42.62
	MHR	45.53	44.62	44.61
	Our Method	**50.57**	**49.74**	**49.76**
T5-XXL	LoRA	49.97	48.89	48.93
	MoE-LoRA	52.14	51.12	51.15
	Poly	55.42	54.65	54.64
	MHR	55.81	55.01	55.01
	Our Method	**62.23**	**61.44**	**61.47**

Table 8: FLAN-T5 with different parameter scales' performances with different adaptation methods on the 100 randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics

Base Model	**PEFT Method**	**Rouge1**	**RougeL**	**RougeLsum**
FLAN-T5-Large	LoRA	68.26	67.42	67.42
	MoE-LoRA	68.59	67.76	67.75
	Poly	68.45	67.60	67.58
	MHR	68.84	67.77	67.78
	Our Method	**68.69**	**67.80**	**67.82**
FLAN-T5-XL	LoRA	71.01	70.21	70.24
	MoE-LoRA	71.08	70.29	70.33
	Poly	71.12	70.31	70.35
	MHR	71.18	70.36	70.40
	Our Method	**71.57**	**70.72**	**70.74**
FLAN-T5-XXL	LoRA	71.89	71.07	71.08
	MoE-LoRA	72.05	71.25	71.26
	Poly	72.55	71.76	71.78
	MHR	72.40	71.61	71.63
	Our Method	**73.09**	**72.27**	**72.28**

A.3 PEFT Experiment with different numbers of Common/task-specific skills

We designed an experiment to investigate the impact of the relative size of common skill and task specific skill on the results under the same parameter quantity. From the table 9, it can be seen that for different base models, the gain is most significant when the number of task specific skills is 1, indicating that explicitly separating the general skills and proprietary skills of skills is the main factor that brings improvement, which is consistent with the methods and viewpoints explained in our paper.

Table 9: Common/task-specific skills experiment on T5-Large, FLAN-T5-Large and GLM-10B with different adaptation methods on the 100 randomly selected tasks from SuperNI dataset. We report the average Rouge1, RougeL, and RougeLsum for all tasks. Higher is better for all metrics

Base Model	N Common Skills	N Specific Skills	Rouge1	RougeL	RougeLsum
T5-Large	4	0	43.04	42.05	42.09
	3	1	49.34	48.50	48.51
	2	2	48.79	47.94	47.94
	1	3	47.46	46.60	46.59
FLAN-T5-Large	4	0	68.45	67.60	67.58
	3	1	68.69	67.80	67.82
	2	2	68.47	67.63	67.65
	1	3	68.21	67.36	67.41
GLM-10B	4	0	47.96	46.80	46.80
	3	1	49.53	48.45	48.45
	2	2	49.19	48.16	48.16
	1	3	46.85	45.74	45.73

A.4 Model Scale Experiment on SuperGLUE

We conduct SuperGLUE experiment on FLAN-T5-XL as shown in Figure 4 and Table 10 to show the performance of different PEFT methods. Experiments have shown that compared to FLAN-T5-Large (0.78B) in Figure 2, FLAN-T5-XL (2B) can achieve a more stable improvement on the SuperGLUE dataset after increasing the parameter quantity, this indicates that as the number of parameters increases, the model's fitting ability becomes stronger and can adapt to the parameter learning needs of multiple tasks, while our method weakens the occurrence of negative transfer phenomenon.

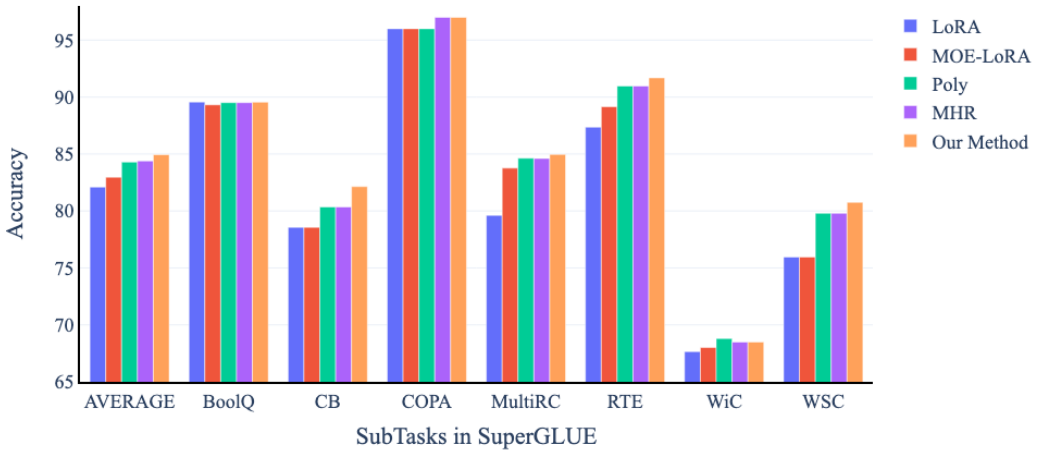

Figure 4: FLAN-T5-XL with different PEFT methods on SuperGLUE benchmark. We reported overall averaged (AVERAGE) and task-specific accuracy for all sub-tasks.

Table 10: FLAN-T5-XL with different adaptation methods on the SuperGLUE dataset. We report the overall (matched and mismatched) accuracy for BoolQ, CB, COPA, MultiRC, RTE, WiC and WSC. Higher is better for all metrics.

	avg	boolq	cb	copa	multirc	rte	wic	wsc.fixed
lora	82.10	**89.57**	78.57	96.00	79.62	87.36	67.65	75.96
moe-lora	82.97	89.32	78.57	96.00	83.76	89.16	68.02	75.96
poly-lora	84.29	89.51	80.35	96.00	84.63	90.97	**68.80**	79.80
mhr-lora	84.39	89.51	80.35	97.00	84.61	90.97	68.49	79.80
cpoly-lora	**84.94**	89.55	**82.14**	**97.00**	**84.96**	**91.69**	68.50	**80.76**

A.5 Learned Common-Skill Router Visualization and Analysis

Figure 5, Figure 6, and Figure 7 shows the distribution of router weights of common skill for selected transformers layers (we select layer0-10-20-30-40 of transformer layers in GLM-10B Model) in NI-10, NI-50, NI-100 Tasks experiment of GLM-10B, corresponding to the three sets of experiments in Appendix: Table 6. The allocation of weights shows that the learned router vector weights for different tasks are clearly separated and differentiated.

Figure 8 and Figure 9 show that our method achieves stronger learning ability in the common-skill router after independently splitting task-specific skills for individual learning, which allows for more discriminative learning of differences and similarities among tasks. This is reflected in the more reasonable types of tasks clustered by our router, with a more balanced number of tasks in each abstract class and more sufficient learning than the origin poly.

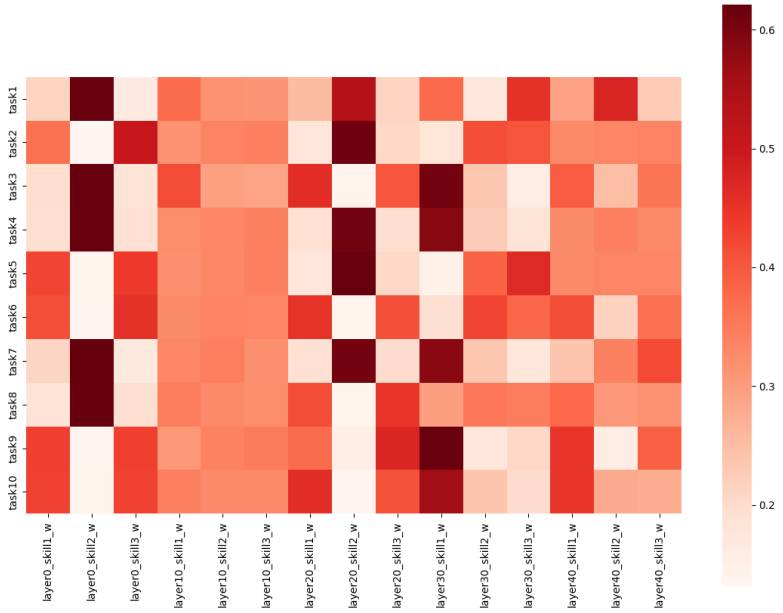

Figure 5: Visualization of the common skill allocation matrix W_A of selected transformer layers in GLM-10B after training on SuperNI-10.

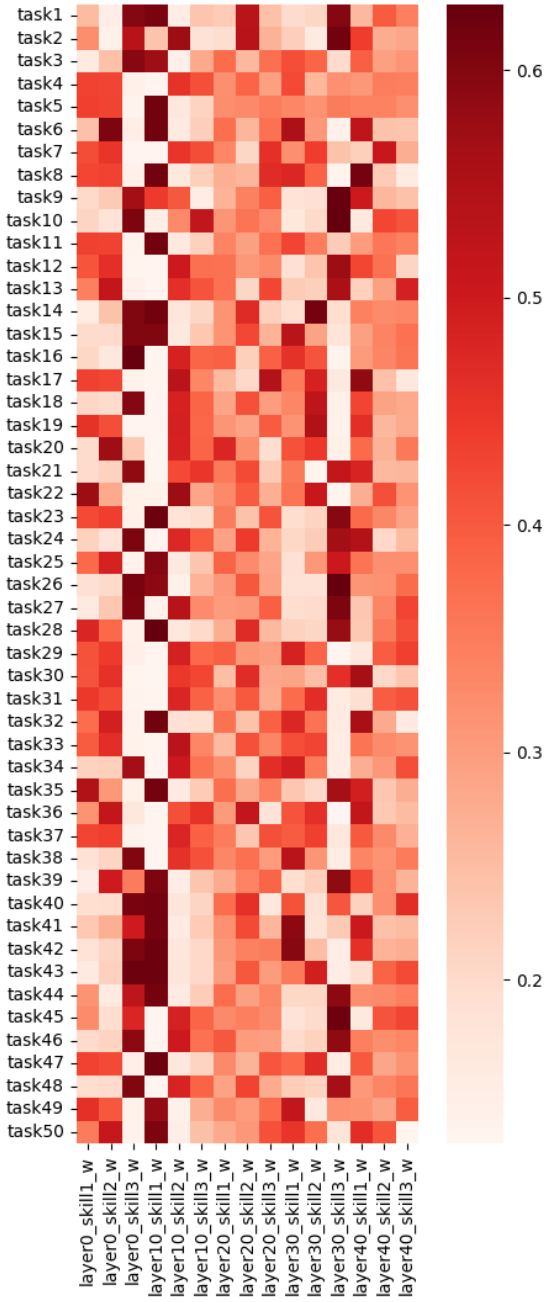

Figure 6: Visualization of the common skill allocation matrix W_A of selected transformer layers in GLM-10B after training on SuperNI-50.

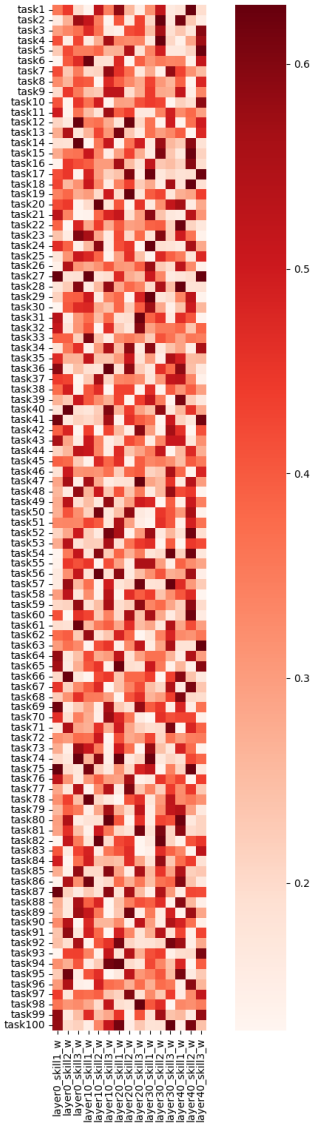

Figure 7: Visualization of the common skill allocation matrix W_A of selected transformer layers in GLM-10B after training on SuperNI-100.

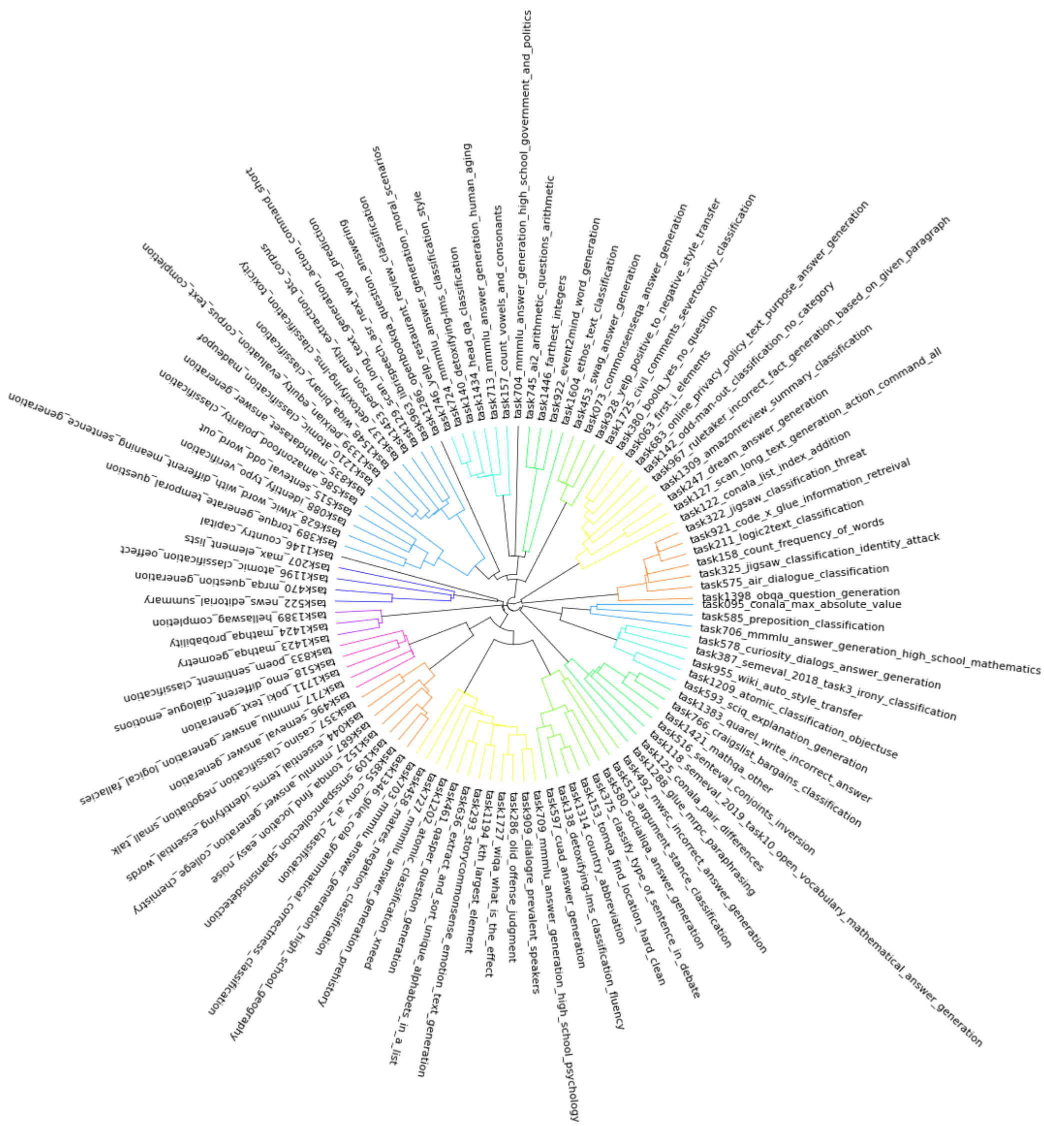

Figure 8: Task clustering dendrogram for common skill allocation matrix W_A of `C-Tuning` using GLM-10B as the base model in the NI-100-Tasks experiment. Tasks are grouped into the same category if they share a similar subset of skills.

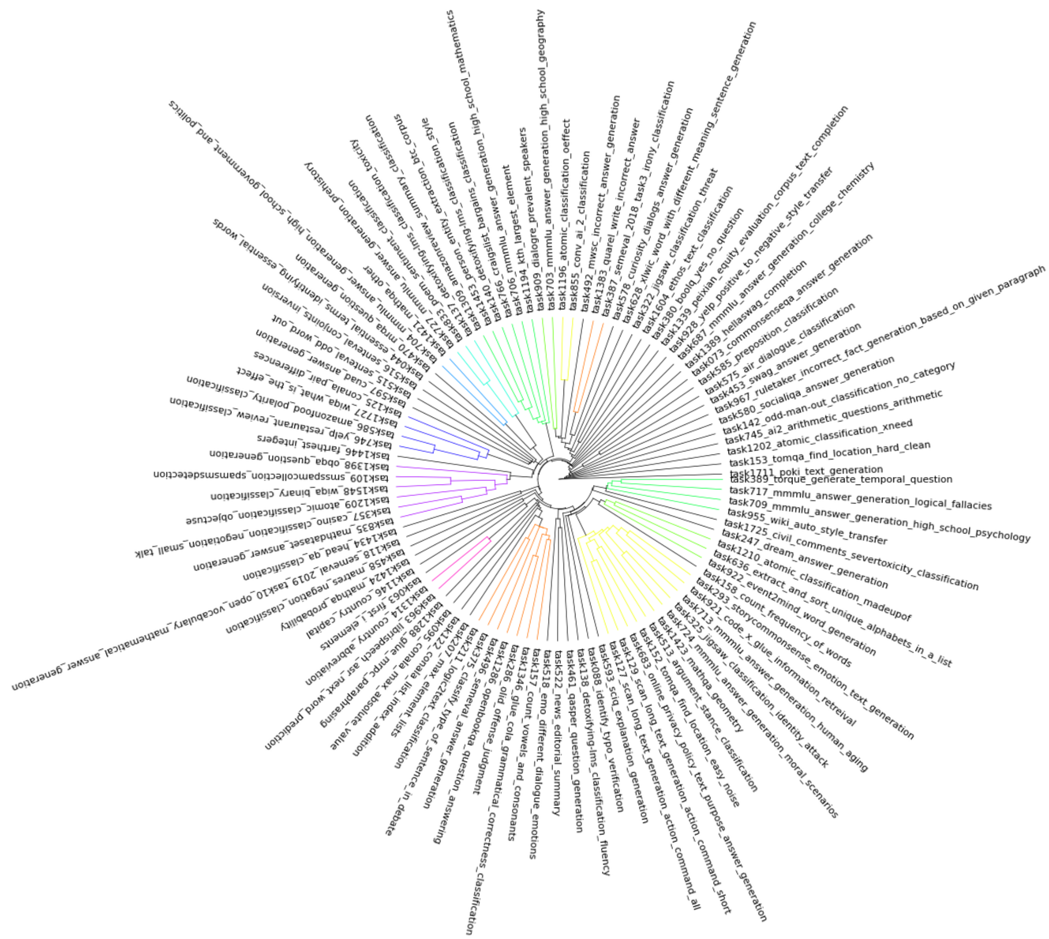

Figure 9: Task clustering dendrogram for common skill allocation matrix W_A of `Poly` using GLM-10B as the base model in NI-100-Tasks experiment. Tasks are grouped into the same category if they share a similar subset of skills.