Let the Expert Stick to His Last: Expert-Specialized Fine-Tuning for Sparse Architectural Large Language Models

Anonymous ACL submission

Abstract

Parameter-efficient fine-tuning (**PEFT**) is crucial for customizing Large Language Models (**LLMs**) with constrained resource. Although there have been various PEFT methods for dense-architecture LLMs, PEFT for sparse-architecture LLMs is still underexplored. In this work, we study the PEFT method for LLMs with the Mixture-of-Experts (**MoE**) architecture and the contents of this work are mainly threefold: (1) We investigate the dispersion degree of the activated experts in customized tasks, and found that the routing distribution for specific task tend to be highly concentrated, while the distribution of activated experts varies significantly across different tasks. (2) We propose the expert-specialized fine-tuning method, which tunes the experts most relevant to downstream tasks while freezing the other experts and modules; experimental results demonstrate that our method not only improves the tuning efficiency, but also matches or even surpasses the performance of full-parameter fine-tuning. (3) We further analyze the impact of the MoE architecture on expert-specialized fine-tuning. We find that MoE models with finer-grained experts are more advantageous in selecting the combination of experts that are most relevant to downstream tasks, thereby enhancing the both the training efficiency and effectiveness.

1 Introduction

As the parameter scale of large language models (**LLMs**) continues to increase (Meta, 2024; Mistral, 2024a; DeepSeek, 2024; Qwen, 2024), parameter-efficient fine-tuning (**PEFT**) methods (Han et al., 2024) are becoming more and more important in adapting pre-trained LLMs to downstream customization tasks. However, existing works (Hu et al., 2021; Liu et al., 2021) on PEFT have primarily focused on dense-architecture LLMs, with research on sparse-architecture LLMs still being markedly insufficient.

In this work, we focus on exploring PEFT techniques within the Mixture-of-Experts (**MoE**) LLMs (Mistral, 2024b; Databricks, 2024). Unlike dense model where all tasks are handled by the same parameters, in the MoE architecture, different tasks are processed by distinct activated experts (Lepikhin et al., 2021; Fedus et al., 2021). Motivated by the observation that specialization of tasks in expert systems is the key to the performance of MoE LLMs (Dai et al., 2024), we propose Expert-Specialized Fine-Tuning (**ESFT**) solution (as shown in Figure 1), which only tunes a limited subset of experts with the highest affinity to the customization task, while freezing the parameters of the other experts and other modules.

The primary advantages of ESFT lie in two aspects: (1) **Saving Computation Resources**: only the parameters of the selected experts need to be updated, which effectively reduces the storage, memory and training time required for tuning. Empirical results indicate that generally selecting less than 25% experts can achieve near-performance in different tasks. (2) **Maintaining Expert Specialization**: ESFT can prevent the decrement of specialization in full-parameter fine-tuning, where experts not adept at the task also update their parameters. Experimental results demonstrate that the ESFT can achieve aligned or even superior performance in downstream task evaluations compared to full-parameter fine-tuning. Additionally, it better maintains performance in general tasks when learning new tasks.

Besides, we delved deeper into the reasons why our method works. We analyze the distribution of activated experts among different tasks. We discover that the distribution of experts activated by the same task's data is quite concentrated, while there are significant differences among the distributions of experts activated by different tasks' data. This analysis indicates that the MoE model utilizes specialized combinations of experts to han-

dle different tasks, and our method can strengthen this tendency toward specialization. In contrast, updating all expert parameters can lead to a reduction in this level of specialization.

More importantly, our investigative experiments reveal that a key factor of our approach is the fine-grained expert system. We take the DeepSeek-V2-Lite (DeepSeek, 2024) as the experiment backbone, which features a much more refined expert division (8 out of 66 experts are activated for each token) compared to other MoE models (Lepikhin et al., 2021; Fedus et al., 2021). The fine-grained MoE model facilitates our approach in selecting the expert combinations that are most relevant to the task, thereby enhancing both the learning efficiency and effectiveness on downstream tasks.

2 Related Work

2.1 Parameter-efficient fine-tuning for dense architectural LLMs

The goal of parameter-efficient fine-tuning (Han et al., 2024) is to efficiently customize LLMs for downstream tasks, while existing studies primarily focused on dense architectural LLMs. PEFT methods for dense models can generally be categorized into three approaches: (1) **Adding new parameters**: methods of this kind fix the existing model parameters and fine-tune the model on a small number of newly-added parameters. Adapter (Houlsby et al., 2019; Pfeiffer et al., 2020; He et al., 2021; Wang et al., 2022) and Soft Prompt (Li and Liang, 2021; Liu et al., 2021; Zhang et al., 2023b; Lester et al., 2021) are two typical representatives of this category of methods. (2) **Selecting existing parameters**: methods of this type fine tune a limited part of existing parameters, while keeping the majority of the other parameters fixed. Based on whether the trainable parameter space is continuous, these methods can generally be divided into structured training (Guo et al., 2020; Gheini et al., 2021; He et al., 2023; Vucetic et al., 2022) and unstructured training (Liao et al., 2023; Ansell et al., 2021; Sung et al., 2021; Xu et al., 2021). (3) **Applying low-rank adaptation**: LoRA (Hu et al., 2021; Fomenko et al., 2024) is a widely-used PEFT method, which decomposes the origin weight matrices into low-rank components. Subsequent works (Zhang et al., 2023a; Ding et al., 2023; Lin et al., 2024; Liu et al., 2023) have introduced numerous improvements to the original LoRA method. However, PEFT study of MoE models is still scarce. In this work, we select and tune part of experts according to their affinity to downstream task, which is a unique selection dimension exclusive to the sparse MoE architecture.

2.2 Coarse- and Fine-grained MoE LLMs

Compared to dense-structure LLMs (e.g. LLaMA series (Meta, 2023b,a)), MoE-structure LLMs (e.g. Mixtral MoE series (Mistral, 2024a,b)) can increase model size while saving on inference costs. Based on the granularity of experts, existing large MoE architectural models can generally be divided into two categories: coarse- and fine-grained experts. Most existing MoE LLMs (Lepikhin et al., 2021; Fedus et al., 2021; Roller et al., 2021; Dai et al., 2022; Shen et al., 2024) have coarse-grained expert systems where the number of experts all very limited. For example, 2 out of 8 experts are activated for Mixtral MoE series (Mistral, 2024a,b) and Grok-V1 (XAI, 2024). As a result, the same expert has to learn complicated patterns from different domain tasks simultaneously. To address this issue, the DeepSeek MoE (Dai et al., 2024) has introduced a fine-grained expert pattern. In the DeepSeek-V2 (DeepSeek, 2024), there are as many as 162 experts, with 8 active experts (8 out of 66 experts are activated for the DeepSeek-V2-Lite). The fine-grained division of experts ensures a high degree of specialization among the experts. Moreover, the specialized expert system enables the selection of experts that are most relevant to the task for efficient tuning.

3 Methods

3.1 Preliminaries: Mixture-of-Experts for Transformers

In the Mixture-of-Experts (MoE) architecture for Transformers, MoE layers can replace Feed-Forward Networks (FFNs). Each MoE layer contains multiple experts structurally identical to a standard FFN. Tokens are assigned to and processed by only a subset of experts based on their affinity scores. The sparse gate routing mechanism ensures computational efficiency in MoE layers, as each token is assigned to a subset of experts.

The output hidden state $\mathbf{h}_t^l$ of the t-th token in the l-th MoE layer is computed as:

$$\mathbf{h}_t^l = \sum_{i=1}^{N} \left(g_{i,t} \mathrm{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l, \qquad (1)$$

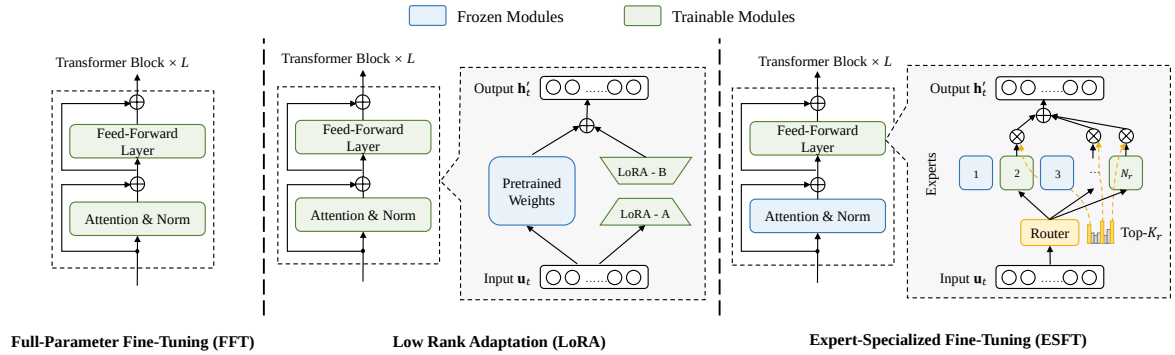

Figure 1: Comparison between Expert-Specialized Fine-Tuning (ESFT) and other fine-tuning methods. FFT trains all parameters. LoRA combines pretrained weights with low-rank matrices to reduce training cost. ESFT only trains a subset of experts in an Mixture-of-Expert (MoE) architecture, optimizing efficiency and task specialization.

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{TopK}(\{s_{j,t} | 1 \leqslant j \leqslant N\}, K), \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

$$s_{i,t} = \text{Softmax}_i \left(\mathbf{u}_t^{l\top} \mathbf{e}_i^l \right), \tag{3}$$

where N denotes the total number of experts, $\text{FFN}_i(\cdot)$ is the i-th expert FFN, $g_{i,t}$ denotes the gate value for the i-th expert, $s_{i,t}$ denotes the token-to-expert affinity, $\text{TopK}(\cdot, K)$ denotes the set comprising K highest affinity scores among those calculated for the t-th token and all N experts, and $\mathbf{e}_i^l$ is the centroid of the i-th expert in the l-th layer.

Recently, DeepseekMoE (Dai et al., 2024) proposed enhancements to the MoE architecture through fine-grained expert segmentation. It segments each expert FFN into multiple smaller experts and keeps the fraction of experts computed, enabling the smaller experts to specialize in different knowledge types while maintaining the same computational cost. Mathematically, the output of an MoE layer with fine-grained segmentation is:

$$\mathbf{h}_t^l = \sum_{i=1}^{mN} \left(g_{i,t} \text{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l, \tag{4}$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{TopK}(\{s_{j,t} | 1 \leqslant j \leqslant mN\}, mK), \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

where each expert is segmented into m small ones.

3.2 Task-Specific *Specialization* in MoE Models

Despite the significant success of MoE LLMs, a clear understanding of the underlying mechanisms remains elusive. We conducted several experiments to understand how experts are selected and utilized across various tasks. These tasks, as detailed in §4.1, include general domains such as math and code, as well as specialized domains like translation, intent recognition, text summarization, and legal judgment prediction. These experiments reveal the concentration and specialization of experts.

Expert Routing is Concentrated in a Task We investigate the distribution of normalized gate values for each expert in various tasks, as shown in Figure 2. Gate values are the sum of all expert-token gate values for each expert, normalized by dividing by the total across all experts. In the figure, the experts are sorted by their normalized values from high to low. The figure shows that a small subset of experts handles the majority of gate values, indicating the model's specialization and efficient expert allocation for specific tasks.

Active Experts Vary Significantly by Task We investigate the joint distribution of experts across tasks. Figure 3 shows a heatmap of the shared TOP-6 routed experts between tasks. The number indicates the shared experts averaged across layers for two independent sets of samples for each task. The off-diagonal values are near zero and the diagonal values are near 6, showing that the same task uses similar sets of experts while different tasks use different sets. Therefore, each task leverages a distinct subset of experts.

3.3 Expert-Specialized Fine-tuning

The highly specialized expert routing suggests that different experts can be optimized for specific tasks. Inspired by this, we propose Expert-Specialized Fine-Tuning (ESFT) that selectively fine-tune the most relevant experts for each task. Our method enhances computational efficiency

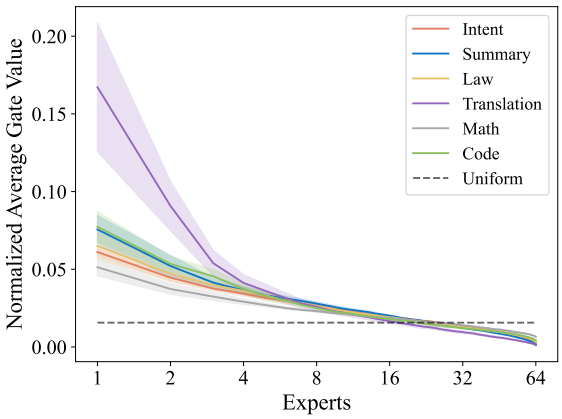

Figure 2: Top Expert distribution for specific tasks. Shaded areas represent variance across layers. The lines show that few experts handle most gate values, highlighting expert specialization for different tasks.

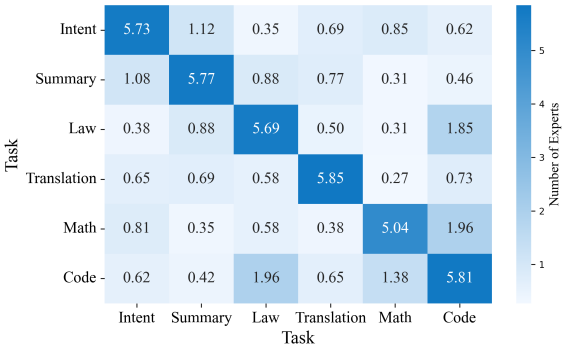

Figure 3: The average number of shared TOP-6 experts across tasks. The values are averaged by layer, indicating that the sets of experts used for the same task are consistent while different tasks are distinct.

and maintains expert specialization as only the most relevant experts are trained. Figure 1 compares our method and existing methods.

Data Sampling We randomly sample a subset $D_s = \{(x_i, y_i)\}_{i=1}^{N_s}$ from the training data $D = \{(x_i, y_i)\}_{i=1}^{N}$ for expert affinity evaluation, where x_i and y_i denote the input and label, respectively. Empirically, we find that a subset of 32 concatenated samples, each with a fixed sequence length of $L = 4096$, is sufficient and robust to select the most relevant experts for specialized tasks.

Expert Relevance Score We propose two methods to calculate the relevance of an expert to a task, based on its affinity to the tokens in the samples:

1. Average Gate Score (ESFT-Gate):

$$g_i^l = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{1}{L_j} \sum_{k=1}^{L_j} g_{i,k}^l, \tag{6}$$

where L_j is the length of the input sequence x_j in the sampled data D_s. This method calculates the average affinity of expert e_i to all tokens in the sampled data.

2. Token Selection Ratio (ESFT-Token):

$$r_i^l = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{1}{L_j} \sum_{k=1}^{L_j} \frac{\mathbb{1}\left(g_{i,k}^l > 0\right)}{K}, \tag{7}$$

Both methods provide a measure of the relevance of each expert to the downstream task, based on the sampled data. The choice between the two methods depends on the specific characteristics of the task and the MoE model.

Expert Selection and Fine-tuning For each MoE layer l, we select a subset of experts to be fine-tuned based on their relevance scores. We define a threshold $p \in (0, 1]$ as a hyperparameter controlling the proportion of total relevance scores to be included in the selected subset. For each layer l, we select a set of top-scored experts E_s^l whose cumulative relevance score exceeds the threshold p, satisfying:

$$\sum_{i \in E_s^l} R_i^l \geqslant p, \tag{8}$$

where R_i^l is the relevance score (either r_i^l or g_i^l) of expert i in layer l.

During fine-tuning, we only update the selected experts E_s^l in each MoE layer l, while freezing the remaining experts and other modules of the model.

4 Experiment Setup

4.1 Main Evaluation

We evaluate our method on two common scenarios: (1) improving the model's **specific ability in a domain** where the model may already have decent performance; (2) adapting the model to a possibly **narrow but unfamiliar downstream task**.

4.1.1 Specialized Ability Improvement

We choose the Math and Code domains to evaluate our method. These domains are suitable as many pre-trained models perform decently, yet there is significant potential for improvement through training. We expect to assess our method's effectiveness through performance gains.

For the Math domain, we use the Metamath dataset (Yu et al., 2023) for training and use GSM8K (Cobbe et al., 2021) and Math (Hendrycks et al., 2021a) for evaluation. For the

Code domain, We train the model on the evol-codealpaca dataset (Chen et al., 2021b) and assess its performance on HumanEval (Chen et al., 2021a) and MBPP (Austin et al., 2021).

4.1.2 Downstream Task Adaptation

We select four diverse tasks. The tasks cover a range of specific abilities that most models can excel at after training but not without training, aiming to show our method's effectiveness through performance gains. The tasks include: (1) Low-resource Translation in the ChrEn dataset (Li et al., 2023), requiring translating the minority Cherokee to English. (2) Text-to-JSON Intent Recognition in the BDCI-21 Smart HCI NLU Challenge[1], which requires converting text instructions into JSON format for home appliances. (3) Text Summarization in the BDCI-21 Summarization Challenge[2], which summarizes customer service call transcripts. (4) Legal judgment Prediction in the the BDCI-21 Law Event Prediction Challenge[3], where the "case description" and "judgment" are repurposed as a legal judgment prediction task. An example for each task is shown in Appendix A.

To measure model performance, for the text-to-JSON task, we calculate the exact match between model output and reference answer; for other tasks, we employ GPT-4 to score model output between 0 and 10 given reference answer[4].

4.2 General Ability Evaluation

To evaluate whether training on new tasks with different methods leads to catastrophic forgetting on existing tasks, we select a wide range of benchmarks to evaluate the general abilities of the models after training with different methods. These benchmarks include CLUEWSC (Xu et al., 2020), TriviaQA (Joshi et al., 2017), IFEval (Smith and Doe, 2021), MMLU (Hendrycks et al., 2021b), CEval (Wang et al., 2021), HellaSwag (Zellers et al., 2019), and ARC (Clark et al., 2018).

4.3 Model and Training Settings

We use the DeepSeek-V2-Lite (DeepSeek, 2024) model as the backbone model for all experiments. The model features a fine-grained set of 66 experts for each of the 26 transformer layers, making it highly suitable for our method which requires expert specialization. We train the model on a carefully curated alignment dataset that excludes math and code data and take the resulting checkpoint as our base model for subsequent experiments. This alignment phase can activate model ability across different domains while forbidding data leakage for math/code evaluation.

We adopt two baselines: Full-Parameter Fine-Tuning (FFT) and Low-Rank Adaptation (LoRA, Hu et al. (2021)). During training, we maintain a 1:1 ratio for alignment data and task-specific data for all methods, which we observe is highly effective for keeping general abilities obtained from the alignment phase. We train all tasks on 2 servers of 8x Nvidia A100 PCIe GPUs.

For hyperparameter settings, all methods use a batch size of 32 and a sequence length of 4096 for training. For every task, we set the maximum steps of training to 500, and evaluate the model every 100 steps. The learning rates are set to 3e-5, 1e-4, and 1e-5 for FFT, LoRA, and ESFT, respectively, based on a hyperparameter search in {1e-5, 3e-5, 1e-4, 3e-4}. The LoRA rank is set to 8 and scaling is set to 2, following (Hu et al., 2021). The threshold p is set to 0.1 for ESFT-Gate and 0.2 for ESFT-Token, respectively. §6.2 shows how we determine the threshold for ESFT.

5 Results

5.1 Benchmark Performance Results

The results in Table 1 show that our method ESFT achieves competitive performance compared to the baselines. As shown in Table 1, ESFT-Token and ESFT-Gate achieve near-best results in domain-specific abilities like Math, and ESFT-Gate achieves the best performance in the Humaneval task. ESFT also excels in specialized tasks, with ESFT-Gate achieving near-best performance in 3 tasks out of 4. Notably, ESFT-Gate's average of 50.4 is competitive compared to FFT's 51.0, slightly better than ESFT-Token's 49.5, and significantly surpasses LoRA's 45.1.

For general ability evaluation, as illustrated in Table 2, ESFT consistently outperforms FFT and LoRA by showing less performance degradation. Notably, ESFT-token performs better than ESFT-gate, with average scores of 61.5 and 60.6, respectively. The results demonstrate a wide range of retention in tasks such as TriviaQA and IFEval, surpassing FFT's 58.8 and LoRA's 59.1. Both

[1] https://www.datafountain.cn/competitions/511
[2] https://www.datafountain.cn/competitions/536
[3] https://www.datafountain.cn/competitions/540
[4] The exact version we use is gpt-4-1106-preview. The evaluation instructions are in Appendix B

	Math Ability		Code Ability		Specialized Tasks				
	MATH	GSM8K	Humaneval	MBPP	Intent	Summary	Law	Translation	Average
Base	19.7	55.9	42.1	44.6	16.8	59.4	17.1	14.5	33.8
FFT	**23.4**	**66.4**	42.1	42.2	**78.8**	**69.4**	47.0	**38.4**	**51.0**
LoRA	20.6	58.9	39.6	**44.8**	67.8	66.4	39.7	23.1	45.1
ESFT-Token (Ours)	22.6	**66.0**	41.5	42.6	75.6	65.4	45.7	36.2	49.5
ESFT-Gate (Ours)	23.2	64.9	**43.3**	41.8	**78.6**	67.2	**49.1**	35.2	**50.4**

Table 1: Main performance comparison across methods and tasks. Best or near-best results are shown in **bold** and second-best results are underlined. Our method ESFT provides a strong balance of performance across diverse tasks, rivaling FFT and surpassing LoRA, particularly in specialized task domains.

	CLUEWSC	TriviaQA	IFEval	MMLU	CEval	HellaSwag	ARC	Average
Base	81.5	67.7	42.5	57.5	59.9	74.0	53.7	62.4
FFT	80.8 ± 0.9	65.9 ± 0.6	34.2 ± 3.3	55.5 ± 0.8	58.8 ± 0.7	67.9 ± 3.0	48.4 ± 1.9	58.8 ± 1.0
LoRA	74.3 ± 6.2	63.4 ± 4.3	38.7 ± 2.0	55.5 ± 1.0	57.0 ± 1.2	**72.8** ± 1.5	51.7 ± 1.8	59.1 ± 2.0
ESFT-Token	80.9 ± 1.7	**66.7** ± 3.5	**40.7** ± 2.6	**57.2** ± 1.0	**59.6** ± 1.5	72.2 ± 7.0	**52.9** ± 3.0	**61.5** ± 2.2
ESFT-Gate	**81.3** ± 0.9	**66.5** ± 1.9	40.2 ± 1.2	**57.0** ± 0.3	**59.5** ± 0.6	68.2 ± 8.0	51.5 ± 2.5	60.6 ± 1.9

Table 2: General ability performance comparison across methods and tasks. The performance for a task is averaged across all training experiments, taking 95% confidence interval. Best or near-best results are shown in **bold**. Our method ESFT consistently achieves good performance among all tasks.

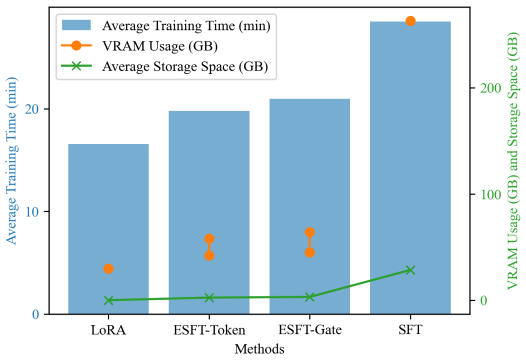

Figure 4: Computational efficiency of different methods. Blue bars show the training time, orange dots/lines indicate VRAM usage and green lines show storage space. ESFT models balance training time, VRAM usage, and storage space efficiently.

methods better retain previously learned knowledge compared to the baselines, highlighting their effectiveness in maintaining task performance.

5.2 Computational Efficiency Results

The results in Figure 6 demonstrates that ESFT exhibits several advantages in terms of training time, VRAM usage, and storage space requirements:

Training Time The average training time for ESFT-Token and ESFT-Gate is 19.8 minutes and 20.9 minutes, respectively. The FFT method takes significantly longer at 28.5 minutes. Although LoRA achieves a shorter training time of 16.5 minutes, our methods are relatively close.

Model VRAM Usage The VRAM usage for ESFT-Token ranges from 42.3 to 58.28 GB across 4 tasks, and for ESFT-Gate from 45.02 to 64.2 GB. These are much lower than the 263 GB required by FFT and comparable to the 30.05 GB used by LoRA. Notably, both our methods and LoRA can train the 16B model on a single A100-80GB GPU, which is not feasible with FFT.

Storage Space The storage requirement (i.e., average storage space of parameters trained) is 2630 MB for ESFT-Token and 3280MB for ESFT-Gate, while FFT demands a substantial 29300 MB. Although LoRA requires less storage of only 107 MB, ESFT offers a more balanced performance in terms of VRAM usage and training time.

In summary, ESFT demonstrates excellent performance in training time, VRAM usage, and storage space requirements. It particularly excels in VRAM and storage space efficiency, significantly outperforming FFT. These advantages show that ESFT can effectively select a subset of experts for better efficiency, making ESFT more competitive and practical for language model customization and efficient adaptation.

6 Analysis

In this section, we investigate the expert selection process of ESFT in §6.1, and demonstrate the per-

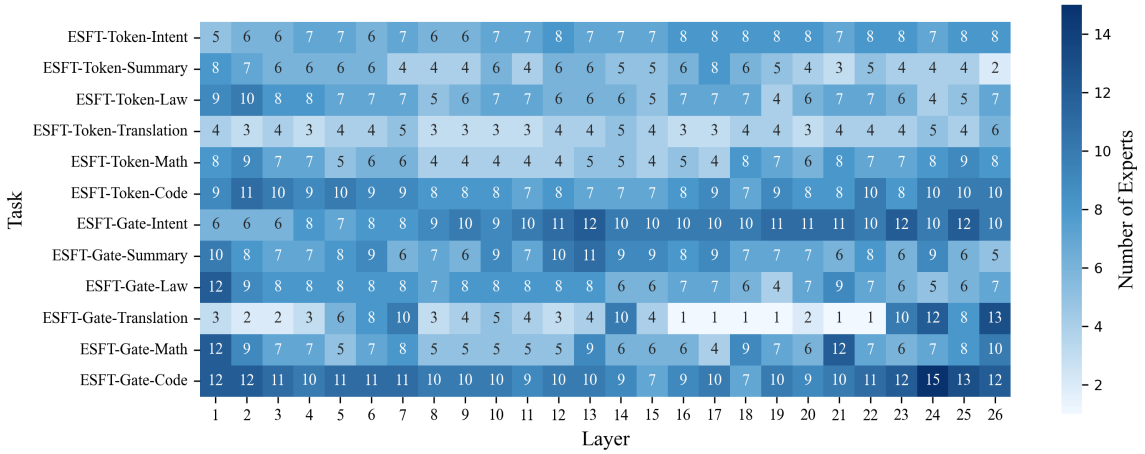

Figure 5: Number of experts trained in ESFT across different layers and tasks. The trained experts are less than 25% of all experts for all tasks, showing the effectiveness of ESFT in selecting the most task-related experts.

formance of ESFT and LoRA under different computational constraints in §6.2. We also conduct ablation experiments in §6.3 to show the importance of our expert relevance scores and fine-grained expert segmentation model architecture.

6.1 ESFT Leverages Specialized Experts Effectively

We analyze the number of experts ESFT trains across tasks and layers to understand its expert selection process. Results are shown in Figure 5.

From the results, we have four key observations: (1) The average number of experts used per task across layers ranges from 2 to 15 out of 66, indicating ESFT can have 75%-95% fewer trainable parameters than FFT; (2) ESFT-Token generally employs fewer experts while better maintaining general performance, comparable to ESFT-Gate in tasks like Math, Intent, and Law; (3) The number of experts varies by task, with more specialized tasks like Math and Translation using fewer experts. Our method's performances for these tasks exceed LoRA to the most extent, indicating that our method is especially suitable for more specialized tasks; (4) For most tasks, few experts are chosen in the middle layers, indicating that expert distribution is more concentrated in these layers.

6.2 ESFT Leverages Training Resources Efficiently

Both ESFT and LoRA have a training efficiency hyperparameter (p for ESFT and rank for LoRA). It affects computational resource usage and potential performance, as a larger value increases computational resource usage and may improve performance. To understand how ESFT and LoRA perform under different efficiency settings, we evaluate benchmark performance on the Math task. We set rank $\leqslant 512$ for LoRA as a higher value will result in more trainable parameters than FFT. Figure 6 illustrates both specialized and general ability under different training efficiency settings.

From the results, we can conclude: (1) All three methods show a trade-off between training efficiency and performance. Increasing trained parameters (p for ESFT and rank for LoRA) can improve performance to a point. (2) ESFT-Token peaks in both specialized and general ability at p=0.5, while ESFT-Gate peaks at p=0.3 for specialized ability and p=0.1 for general ability. (3) ESFT-Token and ESFT-Gate performance saturates at p=0.2 and p=0.1, respectively. (4) Both ESFT-Token and ESFT-Gate outperform LoRA at any point, demonstrating higher specialized ability and more stable general ability. (5) Notably, p=0.2 for ESFT-Token means trained experts cover 20% of expert choices among all tokens, indicating that many task-related tokens, such as punctuation and function words, may be less relevant.

6.3 Ablation Studies

In this section, we demonstrate that the effectiveness of our method lies in two aspects: (1) our proposed expert relevance score function and (2) the fine-grained expert segmentation of the MoE model architecture.

Expert Relevance Score Function In this work, we propose two expert relevance scores: av-

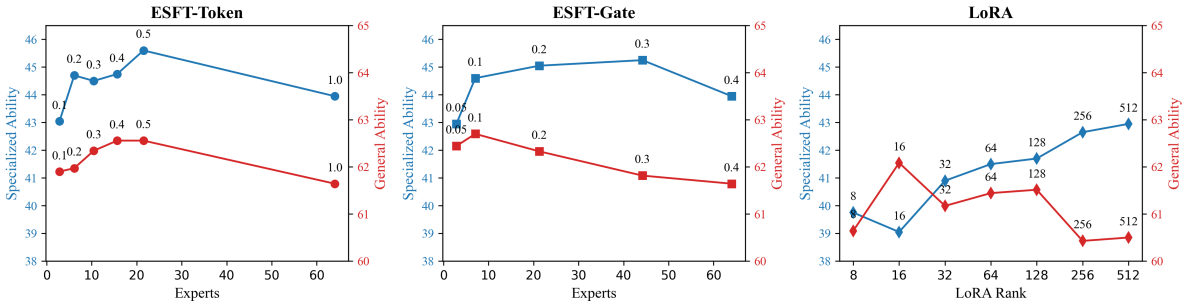

Figure 6: Comparison of three methods under different training efficiency settings on the Math task. The x-axis shows experts for ESFT and rank for LoRA, indicating the ratio of parameters trained. The y-axis represents specialized and general ability. Markers on the lines indicate p and rank values. ESFT-Token and ESFT-Gate consistently outperform LoRA in both specialized and general ability.

	Math Ability		**Code Ability**		**Specialized Tasks**				
	MATH	GSM8K	Humaneval	MBPP	Intent	Summary	Law	Translation	Average
ESFT-Token	22.6	66.0	41.5	42.6	75.6	65.4	45.7	36.2	49.5
Δ of rand	-1.0	-3.7	-2.5	0.2	-2.6	-1.7	1.3	-13.5	-2.9
ESFT-Gate	23.2	64.9	43.3	41.8	78.6	67.2	49.1	35.2	50.4
Δ of rand	-1.7	-3.2	-4.3	1.6	-5.0	-1.1	-2.9	-20.4	-4.6

Table 3: Performance comparison between original experts and random experts. Replacing high-affinity experts with random ones significantly harms model performance across different tasks.

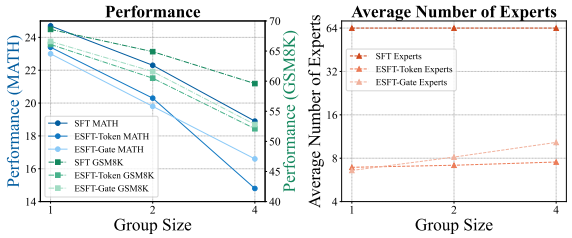

Figure 7: Experiment results for grouped experts. As the experts become more coarse-grained, ESFT degrades more severely than FFT.

erage gate score and token selection ratio, to filter relevant experts for different tasks. To demonstrate their effectiveness, we replace the experts obtained from the two functions with random experts while keeping the number of activated experts for each layer the same. Results in Table 3 show that replacing relevant experts with random ones significantly decreases task performance, demonstrating the effectiveness of our proposed relevance scores.

Fine-Grained Expert Segmentation of the MoE Model We leverage the fine-grained segmented DeepSeek-V2 MoE model as our backbone. To prove the effectiveness of this fine-grained segmentation, we used greedy search (Detailed in Appendix C) to bind experts in groups, simulating models with coarse-grained expert segmentation. Experts in the same group share the same gate for each token, initialized by the average of the original gates' vector. We conduct experiments in the Math domain as an example. Results in Figure 7 show that as the group size increases, the performance of our method decreases more severely than FFT. However, the average number of experts used becomes larger. These observations demonstrate that more fine-grained segmented models will have more specialized experts, making them suitable for our method ans effective LLM customization.

7 Conclusion

In this work, we study parameter-efficient fine-tuning methods for sparse large language models with the Mixture of Experts (MoE) architecture. We observe that tasks from different domains are handled by distinct combinations of experts. We propose selecting the most relevant experts for downstream tasks using two metrics: average gate score and token selection ratio. Experimental results show that our method significantly reduces training costs while matching or surpassing full parameter fine-tuning results. Further analysis confirms that our method enhances the specialization of the expert system within the MoE architecture.

8 Limitation

Firstly, due to the limitation of the availability of other fine-grained MoE models, our method was only tested on the DeepSeek-V2-Lite MoE model. The conclusions drawn from this model require further validation when applied to other contexts. Besides, due to the lack of parameter and structural alignment in MoE models with different expert granularities, we used a simulation approach by binding several groups of experts to compare coarse-grained and fine-grained MoE methods.

References

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. 2021. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint arXiv:2110.07560*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Trevor Cai, Anselm Levskaya, Charles Sutton, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Maarten Dehghani, Pieter Abbeel, Deepak Pathak, Brandon Sanders, Vishal Katarkar, Zareen Xu, et al. 2021a. Evaluating large language models trained on code. In *NeurIPS*.

Mark Chen, Jerry Tworek, Qiming Yuan, Pieter Abbeel, and Deepak Pathak. 2021b. Evol-codealpaca: An evolving dataset for code synthesis. *arXiv preprint arXiv:2108.07740*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Arc: The ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Gsm8k: A dataset for grade school math problem solving. In *NeurIPS*.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *CoRR*, abs/2401.06066.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7085–7095. Association for Computational Linguistics.

Databricks. 2024. Dbrx: Resources and code examples.

DeepSeek. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. *arXiv preprint arXiv:2311.11696*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961.

Vlad Fomenko, Han Yu, Jongho Lee, Stanley Hsieh, and Weizhu Chen. 2024. A note on lora. *arXiv preprint arXiv:2404.05086*.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained transformers for machine translation. *arXiv preprint arXiv:2104.08771*.

Demi Guo, Alexander M Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *CoRR*, abs/2403.14608.

Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. 2023. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11825–11835.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Dan Hendrycks, Collin Burns, Steven Basart, et al. 2021b. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Wei Li, John Smith, and Mary Thompson. 2023. Chren: A cherokee-english low-resource translation dataset. *arXiv preprint arXiv:2304.12345*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. *arXiv preprint arXiv:2305.16742*.

Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, and Hong Mei. 2024. Lora dropout as a sparsity regularizer for overfitting control. *arXiv preprint arXiv:2404.09610*.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Meta. 2023a. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Meta. 2023b. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Meta. 2024. Llama 3 model card.

Mistral. 2024a. Cheaper, better, faster, stronger: Continuing to push the frontier of ai and making it accessible to all.

Mistral. 2024b. Mixtral of experts. *CoRR*, abs/2401.04088.

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

Qwen. 2024. Introducing qwen1.5.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models. *CoRR*, abs/2106.04426.

Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. 2024. Jetmoe: Reaching llama2 performance with 0.1m dollars. *CoRR*, abs/2404.07413.

John Smith and Jane Doe. 2021. Ifeval: A benchmark for evaluating inferential and commonsense reasoning in machines. *Journal of Artificial Intelligence Research*.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205.

Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J Clark, Brett H Meyer, and Warren J Gross. 2022. Efficient fine-tuning of bert models on the edge. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1838–1842. IEEE.

Wei Wang, Yifan Zhang, Qian Liu, et al. 2021. Ceval: A comprehensive evaluation of chinese language models. *arXiv preprint arXiv:2106.02082*.

Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410*, 1(2):4.

XAI. 2024. Grok open release.

Liang Xu, Hai Hu, Xuanwei Zhang, et al. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687*.

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

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

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

Zhen-Ru Zhang, Chuanqi Tan, Haiyang Xu, Chengyu Wang, Jun Huang, and Songfang Huang. 2023b. Towards adaptive prefix tuning for parameter-efficient language model fine-tuning. *arXiv preprint arXiv:2305.15212*.

A Downstream Task Examples

Table 4 presents task descriptions and corresponding responses for different tasks as intent detection, summarization, law prediction, and translation.

B Evaluation Instruction for Benchmarks

Table 5 details the criteria for evaluating tasks such as summary, law prediction, and translation. Each task type includes specific instructions on how to assess predicted answers against reference answers, focusing on aspects such as content accuracy, completeness, relevance, and consistency.

C Strategy for Grouping Experts

In the DeepSeek-V2-Lite architecture, each token selects 6 non-shared experts and two shared experts. These experts selected by the same token are considered similar because they appear together, so we add a score of 1 to them in a co-occurrence matrix until a token sample set (batch size of 32 with sequence length of 4096) is iterated. The similarity between expert i and expert j is calculated using the cosine similarity between the vectors of row i and row j in this co-occurrence matrix.

To obtain an expert grouping strategy through, we calculate the average intra-group similarity (the average pairwise similarity of all experts within the group) for all possible K-expert groups (where K is the group size) from the 64 non-shared experts out of the 66 experts in each layer. We then select a K-expert group with the highest score. For the unselected experts, we repeat this process until all experts are selected and grouped.

Dataset	Task Description
Intent	Prompt: 将二楼主卧的小台灯关掉吧 Response: {”intent”: ”HomeAppliance-Control”, ”slots”: {”appliance”: ” 小台灯”, ”command”: ” 关机”, ”details”: ” 二楼主卧”}}
Summary	Prompt: 【坐席】您好，实习话务员为您服务？【客户】，唉，你好，嗯，请问我这个手机话费这个月为什么这么多呀？【坐席】噢，是这个本月的话费是吧，【客户】嗯，对啊，【坐席】噢，稍等，我帮您看一下啊，女士您好，感谢等待这边看到您这个本月的上网费是产生了169块八毛三，【客户】我我这个上网费就是这个流量用了这么多... Response: 用户来电反馈，对1月份上网费169.83元不认可，我方向用户解释用户强烈不认可，表示并没有收到流量超出提醒短信，用户要求全额退还，烦请后台处理，谢谢！
Law Prediction	Prompt: 1987年4月1日原告**（男方）、被告**（女方）在**民政局登记结婚，1989年2月14日生育女儿**，1990年3月2日生育儿子**，由于二人性格不和，1999年二人分居至今，为此于2016年8月23日诉来**人民法院要求：一、判令原、被告解除婚姻关系；二、本案诉讼费由被告承担。 Response: 依照《中华人民共和国婚姻法》第三十二条第二款、《中华人民共和国民事诉讼法》第六十四条、《最高人民法院关于适用□中华人民共和国民事诉讼法□的解释》第九十条之规定，判决如下：准许原告与被告离婚，本案诉讼费200元，由原告承担。
Translation	ᎠᏁᏄᏗᏍᎨ ᎤᏍᏓᏩᏛᏍᏗ ᎠᏂᏁᎬ ᏧᎵᎢ□ ᎠᏎᏃ ᎨᏍᏗ ᎨᏓᎵ ᎠᎴ ᏕᏚᏏ ᎫᏩᏓᎡᏍᏗ ᏑᎨᏎ ᎢᏧ ᏧᏘ□ ᎨᏍᏗ ᎫᏩᏓᏄᏖᏂᏗ ᏑᎨᏎ ᎠᏓᎾᎾᏛᏍᎬ ᏃᎴ ᎠᎵᏍᎩᏍᎬ□ ᏃᎴ ᏍᏓᏯ ᎠᏦᏑᏢ□ ᎨᏍᏗ ᏅᏍᏓ ᎫᏩᎪᏩᏛᏗ ᏑᎨᏎ ᏂᏕᎵᏍᏔᏂᏙᎲ He tried to follow the instructions his friends were giving him, but he couldn’t run downhill and uphill at the same time, and he couldn’t turn and twist when he was jumping and dancing, and he was crying so hard he could barely see anything that was happening.

Table 4: Task examples for different datasets.

Task Type	Instruction
Summary	请你进行以下电话总结内容的评分。请依据以下标准综合考量，以确定预测答案与标准答案之间的一致性程度。满分为10分，根据预测答案的准确性、完整性和相关性来逐项扣分。请先给每一项打分并给出总分，再给出打分理由。总分为10分减去每一项扣除分数之和，最低可扣到0分。请以 “内容准确性扣x分，详细程度/完整性扣x分，...，总分是：x分” 为开头。1. **内容准确性**：- 预测答案是否准确反映了客户问题或投诉的核心要点。- 是否有任何关键信息被错误陈述或误解。2. **详细程度/完整性**：- 预测答案中包含的细节是否充分，能否覆盖标准答案中所有重要点。- 对于任何遗漏的关键信息，应相应减分。3. **内容冗余度**：- 预测答案是否简洁明了，和标准答案风格一致，不存在冗余信息。- 如果预测答案过长或与标准答案风格不一致，需相应减分。4. **行动指令正确性**：- 预测答案对后续处理的建议或请求是否与标准答案相符。- 如果处理建议发生改变或丢失，需相应减分。预测答案：{prediction} 参考答案：{ground_truth}
Law Prediction	请你进行以下法案判决预测内容的评分。请依据以下标准综合考量，以确定预测答案与标准答案之间的一致性程度。满分为10分，根据预测答案的准确性、完整性和相关性来逐项扣分。请先给每一项打分并给出总分，再给出打分理由。总分为10分减去每一项扣除分数之和，最低可扣到0分。请以 “相关性扣x分，完整性扣x分，...，总分是：x分” 为开头。1. **相关性**：预测答案与标准答案的相关程度是最重要的评分标准。如果预测的判决情况与标准答案完全一致，即所有事实和结果都被精确复制或以不同但等效的方式表述，则应给予高分。若只有部分一致或存在偏差，则根据一致的程度适当扣分。如果没有预测判决内容，扣10分。2. **完整性**：评估预测答案是否涵盖了所有标准答案中提到的关键点，包括但不限于当事人、具体金额、责任判定、费用承担等。如果遗漏重要信息，则应相应扣分。3. **准确性**：检查预测答案中提及的细节、数字、日期和法律依据是否与标准答案保持一致。任何错误信息均需扣分，并且严重错误应该导致更多的扣分。4. **客观性与专业性**：预测答案应客观反映法案内容并使用恰当的法律术语。主观臆断或非专业表达需酌情扣分。预测答案：{prediction} 参考答案：{ground_truth}
Translation	You are an expert master in machine translation. Please score the predicted answer against the standard answer out of 10 points based on the following criteria: Content accuracy: Does the predicted answer accurately reflect the key points of the reference answer? Level of detail/completeness: Does the predicted answer cover all important points from the standard answer? Content redundancy: Is the predicted answer concise and consistent with the style of the standard answer? Respond following the format: ”Content accuracy x points, level of detail/completeness x points, ..., total score: x points”. The total score is the average of all the scores. Do not give reasons for your scores. Predicted answer: {prediction} Reference answer: {ground_truth}

Table 5: Task instructions for model performance evaluation.