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

Zihan Wang[1][2][*], **Deli Chen**[1], **Damai Dai**[1], **Runxin Xu**[1], **Zhuoshu Li**[1], **Yu Wu**[1]
[1]DeepSeek AI
[2]Northwestern University
{zw, victorchen}@deepseek.com

Abstract

Parameter-efficient fine-tuning (**PEFT**) is crucial for customizing Large Language Models (**LLMs**) with constrained resources. 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 a specific task tends to be highly concentrated, while the distribution of activated experts varies significantly across different tasks. (2) We propose **E**xpert-**S**pecialized **F**ine-**T**uning, or ESFT, 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 both the training efficiency and effectiveness. Our code is available at `https://github.com/deepseek-ai/ESFT`.

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 increasingly important in adapting pre-trained LLMs to downstream customization tasks. However, existing works on PEFT like low-rank adaptation (LoRA) and P-Tuning (Hu et al., 2021; Liu et al., 2021) 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), as introduced in §3.1. Unlike dense models 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). Observations indicate that task specialization in expert systems is the key to the MoE LLM performance (Dai et al., 2024). We further illustrate such specialization in §3.2 that experts activated by the same task's data are concentrated, while those for different tasks vary significantly, suggesting MoE models use specialized expert combinations to handle different tasks. Motivated by this, we propose Expert-Specialized Fine-Tuning (**ESFT**), as illustrated in §3.3. ESFT only tunes the experts with the highest affinity to the task, while freezing the parameters of other experts and modules.

The primary advantages of ESFT lie in two aspects: (1) **Maintaining Expert Specialization**: ESFT prevents the decrement of specialization in full-parameter fine-tuning, where experts not adept at the task also update their parameters. Experimental results in §5.1 show that ESFT can achieve aligned or even superior performance in downstream tasks compared to full-parameter fine-tuning, and better maintains performance in general tasks. (2) **Saving Computation Resources**: ESFT only trains the parameters of the selected experts, which effectively reduces the storage of up to 90% and training time up to 30% compared to full-parameter fine-tuning, as shown in §5.2.

Besides, we delve deeper into the working mechanism of the ESFT method. We analyze the expert selection process in §6.1 and demonstrate how

---

[*] Work done during internship at DeepSeek.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 784–801
November 12-16, 2024 ©2024 Association for Computational Linguistics

ESFT leverages specialized experts effectively, as selecting 5-15% experts can achieve promising performance in different tasks. We investigate the efficiency of ESFT under different computational constraints in §6.2, showcasing its ability to leverage training resources efficiently compared to other PEFT methods like LoRA. Our studies in §6.3 analyze the effects of shared and non-shared parameters in the model on specialized and general performance, pointing out the priority to selectively train non-shared parameters in ESFT. Through ablation studies in §6.4, we highlight the importance of our expert relevance scores and the fine-grained expert segmentation architecture.

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 focus 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; Dou et al., 2024) have introduced numerous improvements to the original LoRA method. However, the study of PEFT in sparse models is still scarce. In this work, we select and tune part of the experts based on their downstream task affinity, as a unique selection dimension exclusive to the sparse MoE architecture.

2.2 Coarse- and Fine-grained MoE LLMs

Compared to dense LLMs (e.g., LLaMA series, Meta, 2023b,a), MoE LLMs (e.g., Mixtral series, Mistral, 2024a,b) can increase model size while saving training and inference costs. Based on the granularity of experts, existing large MoE models can generally be divided into two categories: coarse- and fine-grained expert LLMs. 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 experts where the number of experts is 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, a single expert has to learn complicated patterns from different domain tasks simultaneously. To address this issue, DeepSeek MoE (Dai et al., 2024) has introduced fine-grained expert segmentation. 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

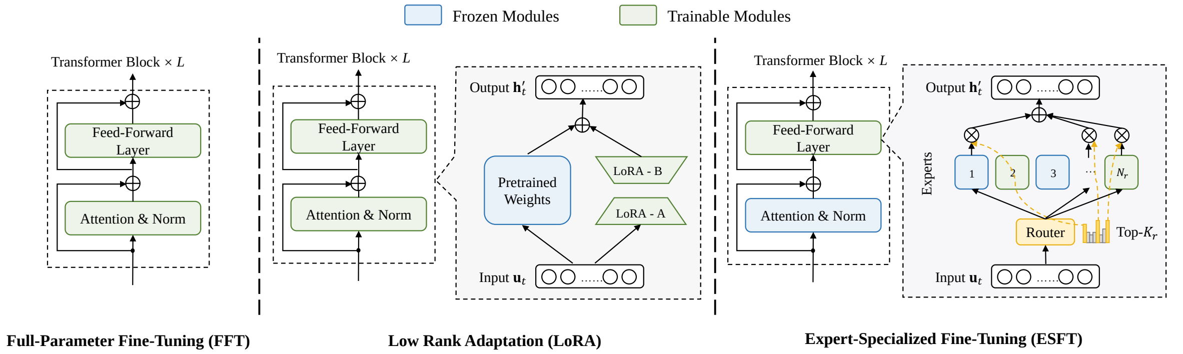
3.1 Preliminaries: Mixture-of-Experts for Transformers

Mixture-of-Experts (MoE) for Transformers replace Feed-Forward Networks (FFNs) with MoE layers. Each MoE layer consists of multiple experts structurally identical to a FFN. Tokens are assigned to and processed by a subset of the most relevant experts based on their affinity scores, ensuring computational efficiency in MoE layers. The output hidden state $\mathbf{h}_t^l$ of the t-th token in the l-th MoE layer is computed as:

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

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

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



Full-Parameter Fine-Tuning (FFT)

Low Rank Adaptation (LoRA)

Expert-Specialized Fine-Tuning (ESFT)

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

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) proposes enhancements to the MoE architecture through several techniques, including (1) Fine-grained segmentation, segmenting each expert into multiple smaller ones and keeping the same fraction of experts to process each token, allowing specialization in different knowledge types while maintaining the same computational cost. (2) Shared expert isolation, leveraging shared experts that process all tokens to capture common knowledge, reducing parameter redundancy and enhancing efficiency. The output of an MoE layer in DeepSeekMoE is:

$$\mathbf{h}_t^l = \sum_{i=1}^{K_s} \text{FFN}_i^s(\mathbf{u}_t^l) + \sum_{i=1}^{N} (g_{i,t} \text{FFN}_i^n(\mathbf{u}_t^l)) + \mathbf{u}_t^l, \tag{4}$$

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

where K_s is the number of shared experts, FFN_i^s and FFN_i^n denote the shared and non-shared experts, respectively. Each expert is segmented into m ones, with N and K also multiplied by m times compared to the coarse-grained architecture.

3.2 Probing Task-Specific Expert *Specialization* in MoE Models

Despite the significant success of MoE LLMs, a clear understanding of the underlying mechanism remains elusive. We conduct probing experiments to understand how non-shared experts are utilized across various tasks. These tasks, as detailed in §4.1, include general domains like math and code, as well as specialized domains like intent recognition, summarization, legal judgment prediction, and translation. These experiments reveal the expert specialization in MoE models in two aspects:

Expert Routing is Concentrated in the Same Task We investigate the distribution of normalized gate values, i.e., the sum of all expert-token gate values for each expert, divided by the total across all experts. Figure 2 displays this distribution, where 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 and concentrated expert allocation for a specific task.

Active Experts Vary Significantly across Tasks We investigate the joint distribution of experts across tasks. Figure 3 shows a heatmap of the shared Top-6 experts for two independent data samples per task averaged across layers. This indicates the degree of overlap of experts used within the same task or between different tasks. Off-diagonal values are near 0, and diagonal values are near 6, indicating that the same task uses similar experts, while different tasks use different sets.

3.3 Expert-Specialized Fine-tuning (ESFT)

The highly specialized expert system suggests that different experts can be optimized for specific tasks. Inspired by this, we propose Expert-Specialized Fine-Tuning (ESFT) for MoE LLM customization, which selectively fine-tunes the most relevant experts for downstream tasks to enhance computa-

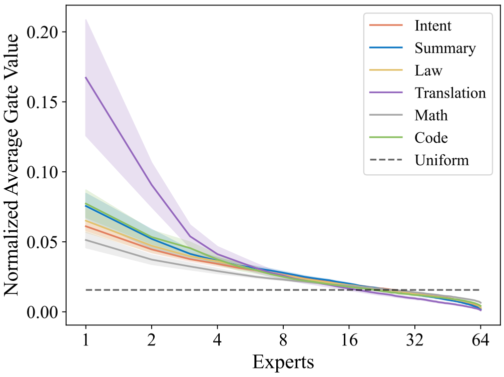
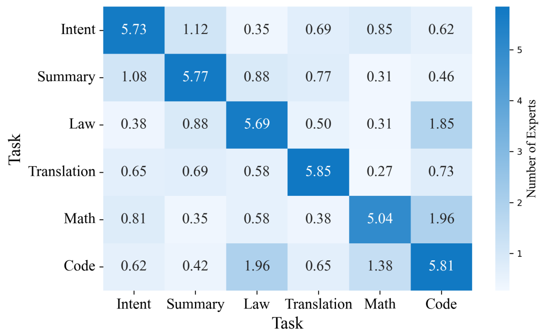


Figure 2: Top Expert distribution for specific tasks. Shaded areas represent variance across layers. The figure shows that few experts handle most gate values, highlighting expert specialization for different tasks.



Task	Intent	Summary	Law	Translation	Math	Code
Intent	5.73	1.12	0.35	0.69	0.85	0.62
Summary	1.08	5.77	0.88	0.77	0.31	0.46
Law	0.38	0.88	5.69	0.50	0.31	1.85
Translation	0.65	0.69	0.58	5.85	0.27	0.73
Math	0.81	0.35	0.58	0.38	5.04	1.96
Code	0.62	0.42	1.96	0.65	1.38	5.81

Figure 3: The average number of shared Top-6 routed 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.

tional efficiency and maintain expert specialization. Figure 1 illustrates the differences between our method and existing methods. Below, we introduce our method step by step.

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 selection, 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 length of $L = 4096$, is robust enough to select the most relevant experts for a task. We detail this claim in Appendix C.

Expert Relevance Score We propose two methods to calculate the relevance of an expert to a task based on its affinity to the sample tokens, defined as average gate score and token selection ratio, respectively. Both methods assess each expert's relevance to downstream tasks and can be chosen based on task-specific experimental performance.

Average Gate Score (ESFT-Gate) This score calculates the average affinity of expert e_i to all tokens in the sampled data. It is defined as:

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

where L_j is the length of the input sequence x_j in the sampled data D_s.

Token Selection Ratio (ESFT-Token) This score calculates the ratio of tokens for which expert e_i is selected. It is defined as:

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

where $\mathbb{1}\left(g_{i,k}^l > 0\right)$ is an indicator that equals 1 if the gate score $g_{i,k}^l$ is positive, and 0 otherwise. K is the number of experts selected per token.

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 minimal 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 training and inference, tokens can be assigned to any expert. However, only the selected experts E_s^l in each layer can be updated; other experts and modules remain frozen.

4 Experiment Setup

4.1 Main Evaluation

We evaluate our ESFT method on two common LLM customization 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 specialized task**.

4.1.1 Tasks for Model Enhancement

We choose two domain-specific tasks, i.e., Math and Code, to evaluate how our method can enhance the model's existing abilities. The two domains are widely concerned in current LLM research and suitable for evaluation, as many pre-trained models can perform decently, while there is significant potential for improvement through further training. We assess our method's effectiveness through performance gains.

For the Math domain, we use MetaMathQA (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 Python subset of the enormous evol-codealpaca dataset (Luo et al., 2023) to simulate a more concentrated LLM customization scenario, and assess its performance on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021).

4.1.2 Tasks for Model Adaptation

We select four specialized tasks to evaluate how our method can facilitate language models to adapt to an unfamiliar downstream task, covering a diverse range of abilities that most models can excel at after training but not without training: (1) 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. (2) Text Summarization in the BDCI-21 Summarization Challenge[2], which summarizes customer service call transcripts. (3) 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. (4) Low-resource Translation in the ChrEn dataset (Zhang et al., 2020), translating the minority Cherokee to English. Examples of the tasks are 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]. All evaluations use few-shot examples.

4.2 General Ability Evaluation

We select a broad range of benchmarks to evaluate the extent to which the models' general abilities are preserved after training on new tasks. These benchmarks include MMLU (Hendrycks et al., 2021b), TriviaQA (Joshi et al., 2017), HellaSwag (Zellers et al., 2019), ARC-Challenge (Clark et al., 2018), IFEval (Zhou et al., 2023), CEval (Huang et al., 2023), and CLUEWSC (Xu et al., 2020), covering comprehensive model ability evaluations across various domains including natural language understanding, question answering, instruction following, and common sense reasoning.

4.3 Backbone Model and Training Settings

We use the backbone architecture of DeepSeek-V2-Lite (DeepSeek, 2024) for all experiments. The model includes a fine-grained set of 66 experts for each transformer layer. This makes it uniquely suitable at the time of this study for our method, which benefits from 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 vanilla model for subsequent experiments. This alignment phase can activate model ability across various domains while keeping Math/Code ability as elementary to better verify the performance gains of our method in these two fields.

We adopt two baselines: Full-Parameter Fine-Tuning (FFT) and Low-Rank Adaptation (LoRA, Hu et al., 2021). For LoRA, we add low-rank matrices to all parameters for training except token embeddings and the language modeling head. We maintain a 1:1 ratio for task-specific data and alignment data for all methods, which we find is highly effective in preserving general abilities obtained from the alignment phase for FFT and LoRA. However, for our ESFT method, not adopting this data mixing strategy may even better maintain general ability. We detail this in Appendix F. All experiments are done on the HFAI cluster[5] with 2 nodes 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.

---

[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 G.

[5] https://doc.hfai.high-flyer.cn/index.html

	Math Ability		**Code Ability**		**Specialized Tasks**				
	MATH	GSM8K	Humaneval	MBPP	Intent	Summary	Law	Translation	Average
Vanilla LM	19.6	55.9	42.1	44.6	16.8	58.6	17.1	14.5	33.6
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	64.7	39.7	23.1	44.9
ESFT-Token (Ours)	22.6	**66.0**	41.5	42.6	75.6	65.4	45.7	36.2	49.4
ESFT-Gate (Ours)	23.2	64.9	**43.3**	41.8	**78.6**	65.8	**49.1**	35.2	50.2

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**
Vanilla LM	81.5	67.7	42.5	57.5	59.9	74.0	53.7	62.4
FFT	80.9 ± 1.1	65.9 ± 0.7	34.2 ± 4.1	55.5 ± 1.0	58.8 ± 0.9	67.9 ± 3.8	48.4 ± 2.4	58.8 ± 1.3
LoRA	74.3 ± 7.7	63.4 ± 5.4	38.7 ± 2.5	55.5 ± 1.2	57.0 ± 1.5	**72.8** ± 1.9	51.8 ± 2.3	59.1 ± 2.5
ESFT-Token	80.9 ± 0.9	**66.7** ± 1.8	**40.7** ± 1.3	**57.1** ± 0.5	**59.6** ± 0.8	72.3 ± 3.6	**52.9** ± 1.5	**61.5** ± 1.1
ESFT-Gate	**81.4** ± 1.1	**66.5** ± 2.3	40.2 ± 1.5	**57.0** ± 0.4	**59.5** ± 0.8	68.2 ± 9.9	51.5 ± 3.1	60.6 ± 2.3

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

5 Results

5.1 Benchmark Performance Results

The results in Table 1 and Table 2 demonstrate several conclusions. All methods can improve model performance in customization tasks compared to the vanilla model, while they may cause a performance decrease in general tasks. Generally, the performance increase is higher in model adaptation tasks than in model enhancement tasks.

For customization ability evaluation, ESFT surpasses LoRA significantly and is competitive with FFT. As shown in Table 1, ESFT-Token and ESFT-Gate achieve near-best results in model enhancement tasks like Math, and ESFT-Gate achieves the best performance in the Humaneval task. ESFT also excels in model adaptation tasks, with ESFT-Gate achieving near-best performance in 3 tasks out of 4. Notably, ESFT-Gate's average of 50.2 is competitive compared to FFT's 51.0, slightly better than ESFT-Token's 49.4, and significantly surpasses LoRA's 44.9. This demonstrates that finding task-relevant experts can efficiently adapt the model for efficient customization.

For general ability evaluation, ESFT consistently outperforms FFT and LoRA by showing less performance degradation. As illustrated in Table 2, 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 methods retain performance better than LoRA and FFT, highlighting their effectiveness in maintaining general task performance[6]. Analyses in §6.3 indicate that such degradation on general tasks for FFT and LoRA may result from training shared parameters.

5.2 Computational Efficiency Results

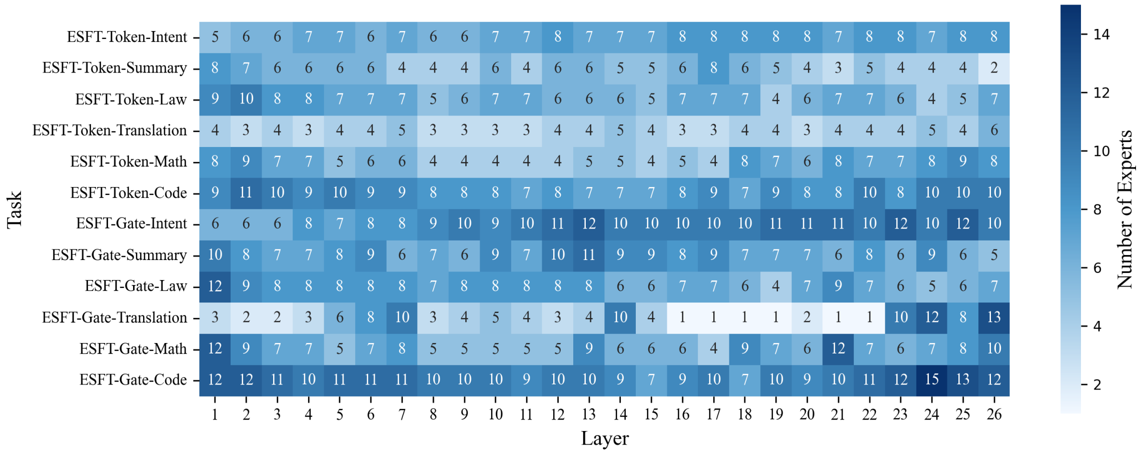
The results in Figure 6 demonstrates that ESFT exhibits several advantages in terms of training time and storage space requirements:

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.

Storage Space The average storage space of parameters trained is 2.57 GB for ESFT-Token and 3.20 GB for ESFT-Gate, while FFT demands a substantial 28.6 GB. Although LoRA requires less storage, ESFT performs significantly better than LoRA in downstream task performance.

---

[6] We further investigate Math and Code performance of the models trained on specialized tasks in Appendix H. FFT and LoRA exhibit even more severe degradation, while ESFT shows a minimal performance drop.



Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
ESFT-Token-Intent	5	6	6	7	7	6	7	6	6	7	7	8	7	7	7	8	8	8	8	8	7	8	8	7	8	8
ESFT-Token-Summary	8	7	6	6	6	6	4	4	4	6	4	6	6	5	5	6	8	6	5	4	3	5	4	4	4	2
ESFT-Token-Law	9	10	8	8	7	7	7	5	6	7	7	6	6	6	5	7	7	7	4	6	7	7	6	4	5	7
ESFT-Token-Translation	4	3	4	3	4	4	5	3	3	3	3	4	4	5	4	3	3	4	4	3	4	4	4	5	4	6
ESFT-Token-Math	8	9	7	7	5	6	6	4	4	4	4	4	5	5	4	5	4	8	7	6	8	7	7	8	9	8
ESFT-Token-Code	9	11	10	9	10	9	9	8	8	8	7	8	7	7	7	8	9	7	9	8	8	10	8	10	10	10
ESFT-Gate-Intent	6	6	6	8	7	8	8	9	10	9	10	11	12	10	10	10	10	10	11	11	11	10	12	10	12	10
ESFT-Gate-Summary	10	8	7	7	8	9	6	7	6	9	7	10	11	9	9	8	9	7	7	7	6	8	6	9	6	5
ESFT-Gate-Law	12	9	8	8	8	8	8	7	8	8	8	8	8	6	6	7	7	6	4	7	9	7	6	5	6	7
ESFT-Gate-Translation	3	2	2	3	6	8	10	3	4	5	4	3	4	10	4	1	1	1	1	2	1	1	10	12	8	13
ESFT-Gate-Math	12	9	7	7	5	7	8	5	5	5	5	5	9	6	6	6	4	9	7	6	12	7	6	7	8	10
ESFT-Gate-Code	12	12	11	10	11	11	11	10	10	10	9	10	10	9	7	9	10	7	10	9	10	11	12	15	13	12

Figure 4: Number of experts trained in ESFT across layers and tasks. Earlier computed layers are numbered smaller. Most tasks and layers train 5-15% of experts, demonstrating ESFT's effectiveness in selecting task-related experts.

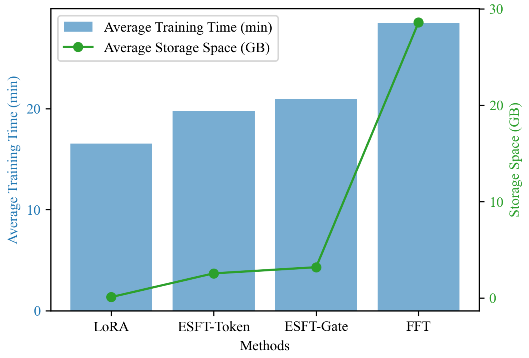


Figure 5: Computational efficiency results. Blue bars show the training time and green lines show storage space, where ESFT both perform high efficiency.

In summary, ESFT demonstrates excellent performance in training time and storage space, significantly outperforming FFT. Furthermore, as shown in Table 3, ESFT requires much fewer trainable parameters compared to FFT, resulting in lower GPU memory usage. These advantages show ESFT's power in efficient language model customization.

6 Analysis

In this section, we investigate the expert selection process of ESFT in §6.1, and demonstrate the performance of ESFT and LoRA under different computational constraints in §6.2. We analyze the effects of training shared and non-shared parameters in §6.3, and conduct ablation studies in §6.4 to verify the importance of our expert relevance scores and model structure of fine-grained experts.

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 4.

From the results, we have several 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 greatest 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). Increasing its value would raise computational resource usage and potentially 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

Non-shared Experts	Shared Experts	Non-expert Parameters	Trainable Parameters	Specialized Ability	General Ability	Average
ALL	✓	✓	15.7B	**51.0**	58.8	54.9
Relevant	✓	×	1.85B	49.8	60.7	55.3
Relevant	×	×	1.4B	49.4	**61.5**	**55.4**
×	✓	×	**450M**	47.4	61.2	54.3
×	✓	✓	1.3B	49.0	60.0	54.5
Relevant	✓	✓	2.7B	**50.8**	60.3	**55.6**
×	×	×	-	33.8	62.4	48.1

Table 3: Comparisons of different model configs based on whether training shared or non-shared parameters. Results include trainable parameters and performance of specialized and general abilities. The best or near-best results excluding the non-training setting are shown in **bold**.

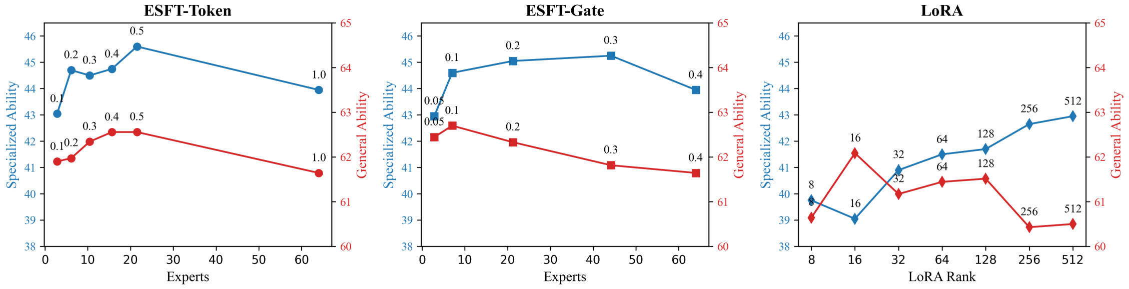


Figure 6: Comparison of three methods under different training efficiency settings on the Math task. The x-axis shows the average trainable experts per layer for ESFT and rank for LoRA, indicating the ratio of trained parameters. The y-axis represents specialized and general ability. Markers on the lines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

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) before a certain point can improve performance. (2) Both ESFT-Token and ESFT-Gate outperform LoRA at any point, demonstrating higher specialized ability and more stable general ability. (3) ESFT-Token peaks in both specialized and general ability at p=0.5, while ESFT-Gate peaks at p=0.3 for specialized and p=0.1 for general ability. (4) ESFT-Token and ESFT-Gate performance saturates at p=0.2 and p=0.1, respectively, indicating that most expert choices may be less relevant to task performance. We delve deeper into this in Appendix E.

6.3 Selectively Training Non-Shared Parameters is the Key to ESFT

In our proposed ESFT method, we only fine-tune a subset of non-shared experts. This section provides detailed discussions of several variants of our method that may also train *shared* parameters. The variables are based on: (1) whether training **non-shared experts** or a task-relevant subset of them (we use the Token Selection Ratio and set p=0.2); (2) whether training **shared experts**; (3) whether training **other shared parameters** including gates, attention layers, and embeddings.

The results are shown in Table 3. We report average trainable parameters across all tasks, performance of specialized and general abilities, and their average. Detailed numbers for all benchmarks are shown in Appendix D. From the results, we can draw several conclusions:

Specialized performance increases as trainable parameters increase. The rank of trainable parameters from 450M to 15.7B highly aligns with the rank of specialized ability from 47.4 to 51.0. This suggests that increasing trainable parameters is effective in enhancing specialized performance.

General performance decreases as trainable *shared* parameters increase. Whether relevant non-shared experts are trained or not, general performance decreases from 61.5 to 60.3, or from 62.4 to 60.0, respectively, as we train shared experts and/or non-expert parameters. As the complete set of non-shared experts is trained, general performance decreases further from 60.3 to 58.8. This suggests that training shared parameters is more

	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	49.4
Δ `of rand`	-1.0	-3.7	-2.5	0.2	-2.6	-1.7	1.3	-13.5	-2.8
ESFT-Gate	23.2	64.9	43.3	41.8	78.6	65.8	49.1	35.2	50.2
Δ `of rand`	-1.7	-3.2	-4.3	1.6	-5.0	0.3	-2.9	-20.4	-4.4

Table 4: Performance comparison between original experts and random experts. Replacing high-affinity experts with random ones significantly harms model performance across different tasks.

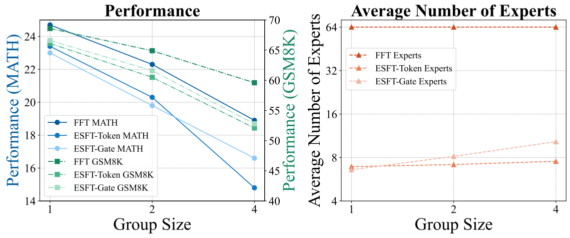


Figure 7: Experiment results for grouped experts. As the experts become more coarse-grained, ESFT degrades more severely than FFT.

likely to cause overfitting and forgetting on general tasks compared to training non-shared parameters.

It is highly prioritized to train task-relevant non-shared experts. Training relevant experts achieves at least 55.3, while other settings achieve at most 54.9, even with higher demands of up to 15.7B parameters. Therefore, fine-tuning these experts is highly prioritized for model customization.

We propose two major training strategies based on these conclusions:

1. **Prioritize specialized ability:** Train all shared parameters and task-relevant non-shared experts to maximize the enhancement of specialized performance.

2. **Balance specialized and general ability, and computational efficiency:** Train only task-relevant non-shared experts to minimize parameter costs while maximizing the maintenance of general ability.

6.4 Analysis of Key Modules in ESFT

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

Expert Relevance Score Function In this work, we propose Average Gate Score and Token Selection Ratio as expert relevance score functions to filter relevant experts for different tasks. To demonstrate their effectiveness, we replace the experts obtained from these functions with random experts while keeping the number of activated experts per layer the same. Results in Table 4 show that replacing relevant experts with random ones significantly decreases task performance, demonstrating the effectiveness of our scoring function.

Fine-Grained Expert Segmentation of the MoE Model We use the fine-grained segmented DeepSeek-V2 model as our backbone. To demonstrate t the effectiveness of this fine-grained segmentation, we use greedy search (as detailed in Appendix B) to group experts, simulating coarse-grained segmentation. Experts in the same group share the average affinity score. We maintain the computational cost by selecting a constant 1/8 of experts for each token. Experiment results of the Math domain in Figure 7 show that as the group size increases, our method's performance decreases more severely than FFT, while the training cost (i.e., trainable experts) rises. These findings indicate that our method, and even effective LLM customization, highly rely on a fine-grained MoE LLM architecture with more specialized experts.

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 first observe that tasks from different domains are handled by distinct combinations of experts. We then 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.

Acknowledgement and Limitations

We would like to thank Xingkai Yu for helping to organize the ESFT open-source training code. 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-wise and structurally aligned 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. 2021. Evaluating large language models trained on code. In *NeurIPS*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/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*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *Preprint*, arXiv:2312.09979.

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*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, et al. 2023. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, arXiv:1705.03551.

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*.

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*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

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.

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.

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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. 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*.

Shiyue Zhang, Benjamin Frey, and Mohit Bansal. 2020. Chren: Cherokee-english machine translation for endangered language revitalization. In *EMNLP2020*.

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*.

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

Appendix

A Examples for Specialized Tasks

Table 5 presents task examples as prompts and corresponding reference responses for each specialized task, including intent recognition, text summarization, legal judgment prediction, and low-resource translation.

B Strategy for Grouping Experts

To group experts together and simulate coarse-grained mixture-of-experts transformer models, we calculate expert similarity and group the experts by maximizing in-group similarities using a greedy search algorithm.

We sample data from the alignment dataset, containing 32 samples each with a sequence length of 4096, to calculate the similarity between experts. We initialize a co-occurrence matrix for all expert pairs as a zero matrix. For each pair of experts that occur simultaneously in a token's Top-6 expert choices, we increment their score by 1 in the matrix. After iterating through the dataset, we calculate the similarity between each pair of experts i and expert j using the cosine similarity between the vectors of row i and row j in the matrix.

To obtain an expert grouping strategy through greedy search, 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, either 2 or 4) from the 64 non-shared experts out of the 66 experts in each layer. We then select the K-expert group with the highest score. For the remaining unselected experts, we repeat this process until all experts are selected and grouped.

C Analysis of Expert Affinity Sample Size

To evaluate the amount of data needed to identify the most relevant experts for a task, we independently sample two sets of data from the training set for each of the six tasks and calculate the shared Top-6 experts between the two sets. The results are shown in Figure 8. As the sample size reaches 2^{17} (i.e., 32 samples with a sequence length of 4096), all tasks exhibit a high number of shared experts between the two samples. This indicates that the sample size is sufficiently large to select the top-relevant experts for the tasks.

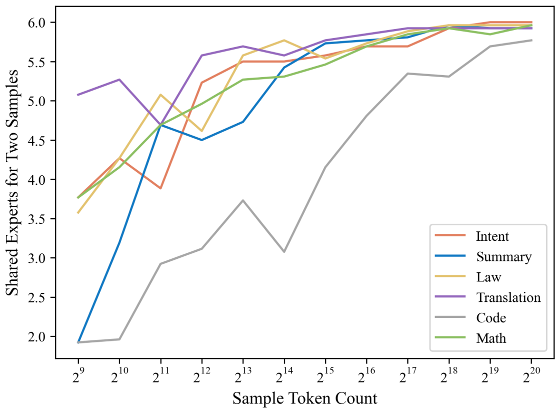


Figure 8: Results of the shared Top-6 routed experts in two independent samples of a task. The x-axis represents the sample size, and the y-axis shows the shared Top-6 routed experts averaged by model layers.

D Detailed Results for Ablations on Training Shared Parameters

We present two tables that summarize the performance of various methods with different configurations for training shared or non-shared parameters. Table 6 shows results on general tasks, and Table 7 focuses on specialized tasks. The results indicate that training only task-relevant non-shared experts consistently maintains the best general task performance. Additionally, training task-relevant non-shared experts and all shared parameters yields the best specialized task performance, short of full-parameter fine-tuning.

E Qualitative Examples of the Expert Choices

We present qualitative examples of the amount that routed experts are trainable among all tokens for each task in Figure 9. Each subfigure demonstrates examples drawn from a task. Deeper tokens indicate more trainable experts across all 26 layers (top-6 experts per layer). The parameter p is set to 0.2 for the token selection ratio. Results show that our method, even handling only about 20% of expert choices, covers a wide range of key task-relevant words.

For example, in the Intent recognition task, the deepest tokens are “意图” (Intent); in the legal judgment task, the deepest tokens include “婚后” (Post-marriage), “要求”(request), “原告” (plaintiff) and “被告” (defendant); in the Math task, the deepest tokens are mainly numerical tokens such as “3”, “5”, “6” and “7”; in the Code task, the deep-

Task	Example
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	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	ᎾᏍᎩ ᎢᏳᏃ ᏌᏉ ᎤᎾᏢ ᏯᎩ-ᎵᏲᎦ, ᏂᎦᏛ ᏄᎾᏢ ᎤᎾᏌᏯᏍᏗᏕ ᎠᏂᎩᎵᏲᎦ; ᎠᎴ ᎢᏳ ᏌᏉ ᎤᎾᏢ ᎦᎸᏉᏔᏅᎯ ᏱᎩ, ᏂᎦᏛ ᏚᎾᏢ ᎤᎾᏌᏯᏍᏗᏕ ᎠᎾᎵᎮᎵᎦ. And whether one member suffereth, all the members suffer with it; or one member is honored, all the members rejoice with it.

Table 5: Examples for different specialized tasks.

Non-shared	Shared	Non-expert	CLUEWSC	TriviaQA	IFEval	MMLU	CEval	HellaSwag	ARC	Average
ALL	✓	✓	**80.9** ± 2.2	65.9 ± 1.5	34.2 ± 8.1	55.5 ± 1.9	58.8 ± 1.7	67.9 ± 7.4	48.4 ± 4.7	58.8 ± 2.5
Relevant	✓	×	**80.9** ± 2.1	66.1 ± 4.4	**42.4** ± 3.0	56.8 ± 1.0	58.9 ± 1.6	67.8 ± 20.4	52.1 ± 5.7	60.7 ± 4.4
Relevant	×	×	**80.9** ± 1.8	**66.7** ± 3.5	40.7 ± 2.6	**57.1** ± 1.0	**59.6** ± 1.5	**72.3** ± 7.0	**52.9** ± 3.0	**61.5** ± 2.3
×	✓	×	**81.1** ± 3.4	**66.7** ± 4.2	41.2 ± 1.6	56.9 ± 1.2	58.9 ± 1.6	71.3 ± 14.1	52.6 ± 5.6	61.2 ± 3.3
×	✓	✓	79.5 ± 4.4	65.8 ± 5.0	41.4 ± 3.2	56.2 ± 1.6	58.6 ± 1.7	67.5 ± 20.7	51.2 ± 4.1	60.0 ± 4.4
Relevant	✓	✓	80.4 ± 4.1	66.3 ± 4.1	41.1 ± 5.0	56.7 ± 1.2	59.0 ± 1.9	67.5 ± 20.3	51.5 ± 4.6	60.3 ± 4.6
×	×	×	81.5	67.7	42.5	57.5	59.9	74.0	53.7	62.4

Table 6: Performance of general tasks across methods based on whether training shared or non-shared parameters. The performance for a task is averaged across all training experiments, followed by the standard deviation across tasks. Best or near-best results are shown in **bold**.

est tokens are key words like “const”, or important commentary words like “Fetch the list of IDs”.

F The Impact of Mixing Alignment Data for Training

We adopt a 1:1 ratio for downstream task data and alignment data for all methods during training to better maintain general task performance. This manual ratio is kept constant to avoid the significant additional costs associated with fine-tuning the ratio for each task.

In this section, we present performance comparisons across various methods and tasks to reveal the impact of mixing alignment data during training. Table 9 presents the performance on downstream specialized tasks, and Table 10 shows the performance on general tasks.

The results indicate that FFT and LoRA benefit from the inclusion of alignment data, leading to improved performance in general tasks while only slightly decreasing performance in downstream tasks. Conversely, our ESFT method does not exhibit the same advantage. Specifically, mixing alignment data does not result in performance increases in either general or downstream tasks. The findings suggest that ESFT is inherently capable of adapting to downstream tasks without significant performance degradation in general tasks, even without added alignment data. This highlights the robustness and adaptability of ESFT in diverse task settings.

G Evaluation Instructions for Specialized Tasks

Table 11 presents the detailed criteria to evaluate specialized tasks including text summarization, legal judgment prediction, and low-resource translation. Each task includes specific instructions on

Non-shared	Shared	Non-expert	Math Ability		Code Ability		Specialized Tasks				
			MATH	GSM8K	Humaneval	MBPP	Intent	Summary	Law	Translation	Average
ALL	✓	✓	23.4	**66.4**	42.1	42.2	78.8	**69.4**	**47.0**	**38.4**	**51.0**
Relevant	✓	×	23.8	65.7	40.2	43.8	80.4	67.3	42.4	35.1	49.8
Relevant	×	×	22.6	66.0	41.5	42.6	75.6	65.4	45.7	36.2	49.4
×	✓	×	22.7	64.5	37.2	44.0	73.6	68.3	42.7	26.0	47.4
×	✓	✓	23.4	**66.6**	41.5	44.4	81.0	66.7	39.0	29.5	49.0
Relevant	✓	✓	**24.8**	66.0	42.1	43.2	**82.2**	**69.5**	46.4	32.2	**50.8**
×	×	×	19.6	55.9	42.1	**44.6**	16.8	58.6	17.1	14.5	33.6

Table 7: Performance of specialized tasks across methods based on whether training shared or non-shared parameters. Best or near-best results are shown in **bold**.

	Math Ability		Code Ability		
	MATH	GSM8K	HumanEval	MBPP	Average
Vanilla LM	19.6	55.9	42.1	44.6	40.5
FFT	15.1 ± 0.3	40.3 ± 5.3	30.2 ± 4.4	40.6 ± 3.9	31.5 ± 2.5
LoRA	11.8 ± 0.6	36.1 ± 4.4	27.9 ± 2.3	36.6 ± 2.6	28.1 ± 2.0
ESFT-Token	**19.4** ± 0.8	**55.2** ± 0.7	**39.5** ± 1.0	44.8 ± 0.8	**39.7** ± 0.4
ESFT-Gate	**19.5** ± 0.3	**55.1** ± 1.3	**39.3** ± 1.3	**45.3** ± 0.6	**39.8** ± 0.6

Table 8: Math and Code performance comparison across methods trained on specialized tasks. Best or near-best results are shown in **bold**. ESFT retains performance significantly better compared to FFT and LoRA.

assessing predicted answers against reference answers, focusing on aspects such as content accuracy, completeness, relevance, and consistency.

H Evaluating Math and Code as General Tasks

We investigate the Math and Code performance of models trained on adaptation tasks (i.e., Intent, Summary, Law, Translation), as these domains reflect the model's general ability if not specifically trained on them. We report numbers with the setting of training on only downstream task data. Results in Table 8 show that FFT and LoRA would lead to significant performance drops in the Math and Code domain, having average performance drops of 9.0 and 12.4, respectively. Notably, our ESFT method retains performance significantly better compared to FFT and LoRA, with an average performance drop of less than 1.0.

	Math Ability		**Code Ability**		**Specialized Tasks**				
	MATH	GSM8K	HumanEval	MBPP	Intent	Service	Law	Translation	Average
FFT	26.1	70.4	51.2	42.6	78.8	72.8	45.6	34.4	52.7
`+ mix data`	-2.7	-4.0	-9.1	-0.4	0.0	-3.4	1.4	4.0	-1.7
LoRA	21.8	57.8	42.1	42.6	78.2	66.4	46.0	21.8	47.1
`+ mix data`	-1.2	1.1	-2.5	2.2	-10.4	-1.7	-6.3	1.3	-2.2
ESFT-Token	25.2	64.8	42.1	43.8	78.0	67.4	47.2	31.9	50.0
`+ mix data`	-2.6	1.2	-0.6	-1.2	-2.4	-2.0	-1.5	4.3	-0.6
ESFT-Gate	24.1	64.9	42.1	44.6	77.2	68.4	43.6	32.8	49.7
`+ mix data`	-0.9	0.0	0.0	-2.8	1.4	-2.6	0.9	2.4	0.5

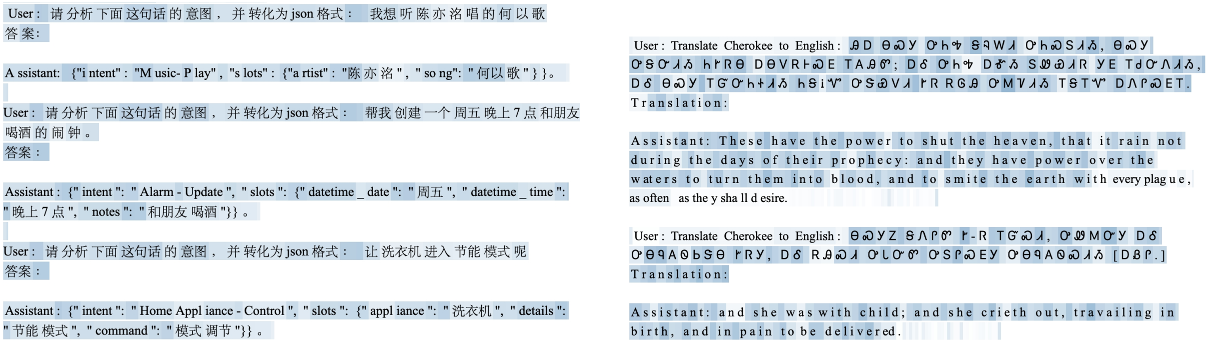
Table 9: Downstream task performance comparison across methods and tasks with and without mixing data from the alignment phase. Results show that mixing alignment data leads to a minor performance decrease for most methods.

	CLUEWSC	**TriviaQA**	**IFEval**	**MMLU**	**CEval**	**HellaSwag**	**ARC**	**Average**
Vanilla LM	81.5	67.7	42.5	57.5	59.9	74.0	53.7	62.4
FFT	76.8 ± 1.7	62.4 ± 10	28.4 ± 5.1	55.5 ± 1.1	58.4 ± 0.4	74.6 ± 3.2	53.6 ± 3.1	58.5 ± 2.5
`+ mix data`	4.1	3.5	5.8	0.0	0.4	-6.7	-5.2	0.3
LoRA	60.2 ± 27	61.2 ± 4.0	33.4 ± 6.1	52.3 ± 3.3	55.3 ± 2.3	71.5 ± 2.5	50.7 ± 2.2	55.0 ± 4.6
`+ mix data`	14.1	2.2	5.3	3.2	1.7	1.3	1.1	4.1
ESFT-Token	80.0 ± 2.5	67.5 ± 0.3	41.9 ± 0.8	57.3 ± 0.2	60.2 ± 0.5	74.5 ± 0.7	54.9 ± 0.7	62.3 ± 0.5
`+ mix data`	0.9	-0.8	-1.2	-0.2	-0.6	-2.2	-2.0	-0.8
ESFT-Gate	80.2 ± 1.6	67.6 ± 0.3	40.8 ± 2.4	57.3 ± 0.3	59.9 ± 0.4	74.3 ± 0.9	55.1 ± 0.9	62.2 ± 0.5
`+ mix data`	1.2	-1.1	-0.6	-0.3	-0.4	-6.1	-3.6	-1.6

Table 10: General task performance comparison across methods and tasks with and without alignment data mixing. Results show that mixing alignment data improves FFT and LoRA in general tasks, but not our ESFT method. It showcases that ESFT can adapt to downstream tasks directly with minimal performance loss in general tasks.

Task	Evaluation 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	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 11: Task instructions for model performance evaluation. The placeholder {prediction} and {ground_truth} represent model prediction and reference answer, respectively.



User：请分析下面这句话的意图，并转化为json格式：我想听陈亦洺唱的何以歌
答案：

A ssistant: {"i ntent" : "M usic- P lay" , "s lots" : {"a rtist" : "陈亦洺 " , " so ng": " 何以歌 " } }。

User：请分析下面这句话的意图，并转化为json格式：帮我创建一个周五晚上7点和朋友喝酒的闹钟。
答案：

Assistant : {" intent ": " Alarm - Update ", " slots ": {" datetime _ date ": " 周五 ", " datetime _ time ": " 晚上7点 ", " notes ": " 和朋友喝酒 "}}。

User：请分析下面这句话的意图，并转化为json格式：让洗衣机进入节能模式呢
答案：

Assistant : {" intent ": " Home Appl iance - Control ", " slots ": {" appl iance ": " 洗衣机 ", " details ": " 节能模式 ", " command ": " 模式调节 "}}。

(a) Intent recognition

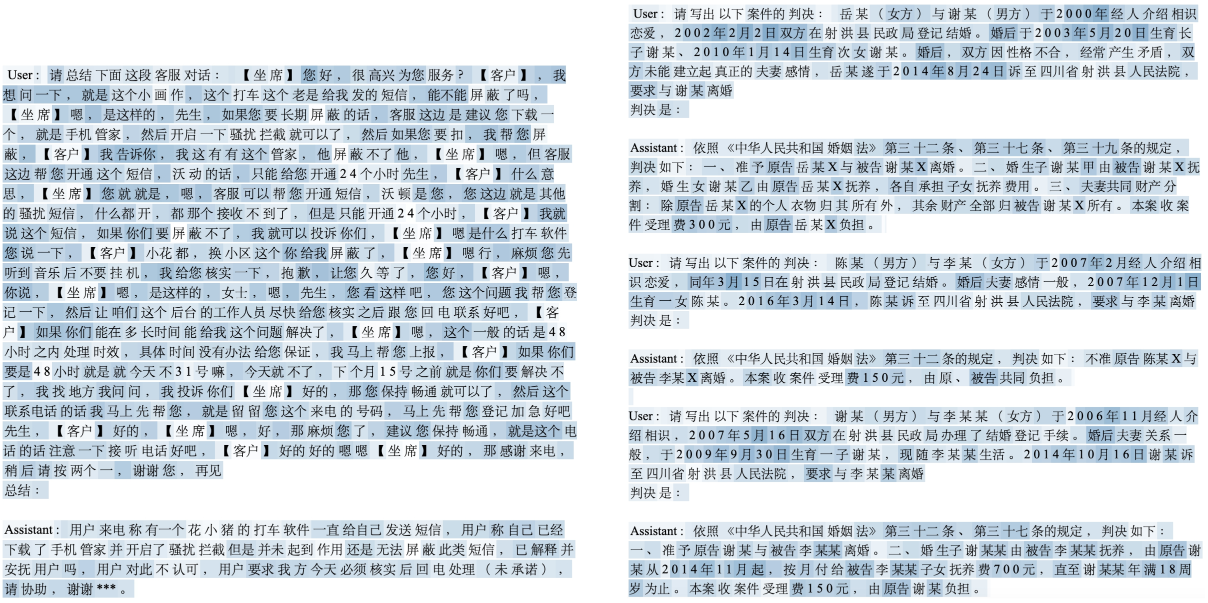
User : Translate Cherokee to English : ᎯᎠ ᎾᏍᎩ ᎤᏂᎲ ᎦᎸᎳᏗ ᎤᏂᏍᏚᏗᏱ, ᎾᏍᎩ ᎤᎦᏅᏗᏱ ᏂᎨᏒᎾ ᎠᎾᏙᎡᎰᏍᎬ ᎢᎪᎯᏛ; ᎠᎴ ᎤᏂᎲ ᎠᎹᏱ ᏚᏪᏬᏗᏒ ᎩᎬ ᎢᏧᏅᏁᏗᏱ, ᎠᎴ ᎾᏍᎩ ᎢᏳᎤᏂᏐᏗᏱ ᏂᎦᎥᏉ ᎤᏚᏬᏙᏗ ᎨᏒ ᎡᎶᎯ ᎤᎷᏤᏗᏱ ᎢᎦᎢᏉ ᎠᏁᎵᏍᎬᎢ.
Translation:

Assistant: These have the power to shut the heaven, that it rain not during the days of their prophecy: and they have power over the waters to turn them into blood, and to smite the earth with every plag u e, as often as the y sha ll d esire.

User : Translate Cherokee to English : ᎾᏍᎩᏃ ᎦᏁᎵᏛ Ꭸ-Ꭱ ᎢᏳᏍᏗ, ᎤᏪᎷᏅᎩ ᎠᎴ ᎤᎾᏄᎪᏫᏏᎦᎾ ᎨᏒᎩ, ᎠᎴ ᎡᎯᏍᏗ ᎤᏓᏅᏛ ᎤᏚᎵᏍᎬᎩ ᎤᎾᏄᎪᏫᏍᏗᏱ [ᎠᏰᎵ.]
Translation:

Assistant: and she was with child; and she crieth out, travailing in birth, and in pain to be delivered.

(b) Low-resource translation



User：请总结下面这段客服对话：【坐席】您好，很高兴为您服务？【客户】，我想问一下，就是这个小画作，这个打车这个老是给我发的短信，能不能屏蔽了吗，【坐席】嗯，是这样的，先生，如果您要长期屏蔽的话，客服这边是建议您下载一个，就是手机管家，然后开启一下骚扰拦截就可以了，然后如果您要扣，我帮您屏蔽，【客户】我告诉你，我这有有这个管家，他屏蔽不了他，【坐席】嗯，但客服这边帮您开通这个短信，沃动的话，只能给您开通24个小时先生，【客户】什么意思，【坐席】您就就是，嗯，客服可以帮您开通短信，沃顿是您，您这边就是其他的骚扰短信，什么都开，都那个接收不到了，但是只能开通24个小时，【客户】我就说这个短信，如果你们要屏蔽不了，我就可以投诉你们，【坐席】嗯是什么打车软件您说一下，【客户】小花都，换小区这个你给我屏蔽了，【坐席】嗯行，麻烦您先听到音乐后不要挂机，我给您核实一下，抱歉，让您久等了，您好，【客户】嗯，你说，【坐席】嗯，是这样的，女士，嗯，先生，您看这样吧，您这个问题我帮您登记一下，然后让咱们这个后台的工作人员尽快给您核实之后跟您回电联系好吧，【客户】如果你们能在多长时间能给我这个问题解决了，【坐席】嗯，这个一般的话是48小时之内处理时效，具体时间没有办法给您保证，我马上帮您上报，【客户】如果你们要是48小时就是就今天不31号嘛，今天就不了，下个月15号之前就是你们要解决不了，我找地方我问问，我投诉你们【坐席】好的，那您保持畅通就可以了，然后这个联系电话的话我马上先帮您，就是留留您这个来电的号码，马上先帮您登记加急好吧先生，【客户】好的，【坐席】嗯，好，那麻烦您了，建议您保持畅通，就是这个电话的话注意一下接听电话好吧，【客户】好的好的嗯嗯【坐席】好的，那感谢来电，稍后请按两个一，谢谢您，再见
总结：

Assistant：用户来电称有一个花小猪的打车软件一直给自己发送短信，用户称自己已经下载了手机管家并开启了骚扰拦截但是并未起到作用还是无法屏蔽此类短信，已解释并安抚用户吗，用户对此不认可，用户要求我方今天必须核实后回电处理（未承诺），请协助，谢谢***。

(c) Text summarization

User：请写出以下案件的判决：岳某（女方）与谢某（男方）于2000年经人介绍相识恋爱，2002年2月2日双方在射洪县民政局登记结婚。婚后于2003年5月20日生育长子谢某、2010年1月14日生育次女谢某。婚后，双方因性格不合，经常产生矛盾，双方未能建立起真正的夫妻感情，岳某遂于2014年8月24日诉至四川省射洪县人民法院，要求与谢某离婚
判决是：

Assistant：依照《中华人民共和国婚姻法》第三十二条、第三十七条、第三十九条的规定，判决如下：一、准予原告岳某X与被告谢某X离婚。二、婚生子谢某甲由被告谢某X抚养，婚生女谢某乙由原告岳某X抚养，各自承担子女抚养费用。三、夫妻共同财产分割：除原告岳某X的个人衣物归其所有外，其余财产全部归被告谢某X所有。本案收案件受理费300元，由原告岳某X负担。

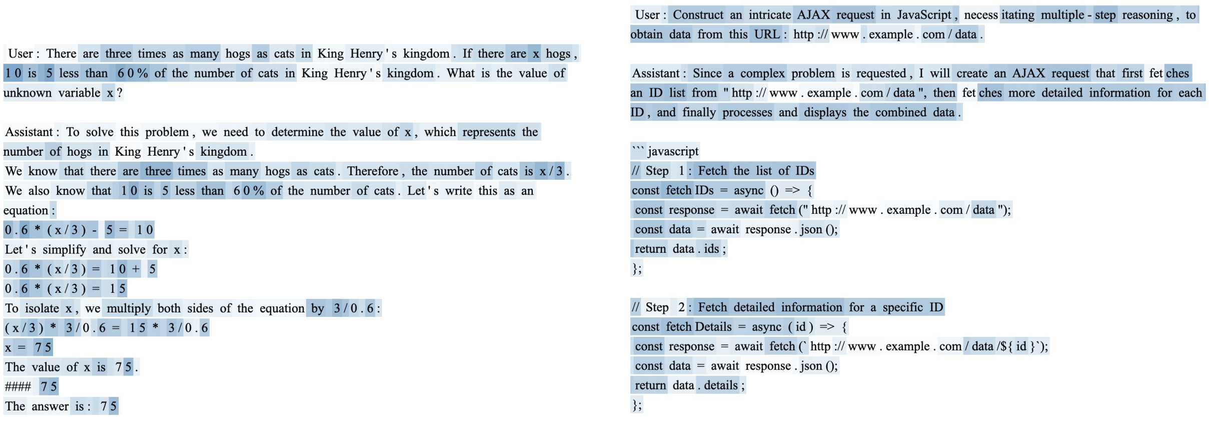
User：请写出以下案件的判决：陈某（男方）与李某（女方）于2007年2月经人介绍相识恋爱，同年3月15日在射洪县民政局登记结婚。婚后夫妻感情一般，2007年12月1日生育一女陈某。2016年3月14日，陈某诉至四川省射洪县人民法院，要求与李某离婚
判决是：

Assistant：依照《中华人民共和国婚姻法》第三十二条的规定，判决如下：不准原告陈某X与被告李某X离婚。本案收案件受理费150元，由原、被告共同负担。

User：请写出以下案件的判决：谢某（男方）与李某某（女方）于2006年11月经人介绍相识，2007年5月16日双方在射洪县民政局办理了结婚登记手续。婚后夫妻关系一般，于2009年9月30日生育一子谢某，现随李某某生活。2014年10月16日谢某诉至四川省射洪县人民法院，要求与李某某离婚
判决是：

Assistant：依照《中华人民共和国婚姻法》第三十二条、第三十七条的规定，判决如下：一、准予原告谢某与被告李某某离婚。二、婚生子谢某某由被告李某某抚养，由原告谢某从2014年11月起，按月付给被告李某某子女抚养费700元，直至谢某某年满18周岁为止。本案收案件受理费150元，由原告谢某负担。

(d) Legal judgment prediction



User : There are three times as many hogs as cats in King Henry ' s kingdom . If there are x hogs , 10 is 5 less than 60 % of the number of cats in King Henry ' s kingdom . What is the value of unknown variable x ?

Assistant : To solve this problem , we need to determine the value of x , which represents the number of hogs in King Henry ' s kingdom .
We know that there are three times as many hogs as cats . Therefore , the number of cats is x / 3 .
We also know that 10 is 5 less than 60 % of the number of cats . Let ' s write this as an equation :
0 . 6 * (x / 3) - 5 = 10
Let ' s simplify and solve for x :
0 . 6 * (x / 3) = 10 + 5
0 . 6 * (x / 3) = 15
To isolate x , we multiply both sides of the equation by 3 / 0 . 6 :
(x / 3) * 3 / 0 . 6 = 15 * 3 / 0 . 6
x = 75
The value of x is 75 .
75
The answer is : 75

(e) Math domain

User : Construct an intricate AJAX request in JavaScript , necess itating multiple - step reasoning , to obtain data from this URL : http :// www . example . com / data .

Assistant : Since a complex problem is requested , I will create an AJAX request that first fet ches an ID list from " http :// www . example . com / data ", then fet ches more detailed information for each ID , and finally processes and displays the combined data .

~~~~
``` javascript
// Step 1 : Fetch the list of IDs
const fetch IDs = async () => {
 const response = await fetch (" http :// www . example . com / data ");
 const data = await response . json ();
 return data . ids ;
};

// Step 2 : Fetch detailed information for a specific ID
const fetch Details = async ( id ) => {
 const response = await fetch (` http :// www . example . com / data /${ id }`);
 const data = await response . json ();
 return data . details ;
};
~~~~

(f) Code domain

Figure 9: Examples for our ESFT method showing the proportion of trainable routed experts among all tokens for each task. Deeper tokens indicate more trainable experts across all 26 layers (top-6 experts per layer). The parameter p is set to 0.2 for the token selection ratio. Results show that our method, even handling only about 20% of expert choices, covers a wide range of key task-relevant words.