Neuronal Redundancy Reduction and Collaborative Learning for Parameter-Efficient Mixture-of-Experts

Anonymous ACL submission

Abstract

Mixture of Experts (MoE) has great potential in scaling up the capacity of large models while maintaining low computational costs. However, existing methods mainly reduce redundancy at the expert level while overlooking fine-grained neuron-level redundancy and expert collaboration. In this paper, we propose an Information Bottleneck based MoE (IBMoE) for parameter-efficient fine-tuning, which reduces neuron-level redundancy within each expert and fosters internal collaboration among all experts. Specifically, IBMoE equips each expert with learnable queries to capture task-specific characteristics and employs sparse neuronal activation to dynamically select the most relevant neurons, reducing redundant activations across tasks. In addition, a diversity constraint is imposed among experts, which maximizes the knowledge difference to enable all experts cooperative more efficiently. Extensive experiments across multiple benchmarks show that IBMoE consistently outperforms strong baselines while activating significantly fewer parameters. In addition, IBMoE reduces inference latency by 63% and memory consumption by 48.5% compared with prior MoE-based approaches, with comparable training cost.

1 Introduction

Large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023; Guo et al., 2025) have achieved strong performance across a wide range of Natural Language Processing (NLP) tasks (Tang et al., 2025; Bai et al., 2025; Aggarwal et al., 2026). However, adapting LLMs to diverse downstream tasks often requires parameter updates, leading to substantial training and inference costs. This motivates parameter-efficient fine-tuning (PEFT) methods, which freeze backbone models and optimize a small set of trainable parameters.

Early PEFT approaches such as prompt tuning (Wang et al., 2025), prefix tuning (Zhao et al., 2025), and P-tuning (Liu et al., 2021) introduce learnable vectors into model inputs or intermediate layers. While effective in some settings, they often underperform on smaller models or more complex tasks. Low-Rank Adaptation (LoRA) (Hu et al., 2021) improves efficiency by learning low-rank updates to model weights, achieving a better trade-off between performance and cost.

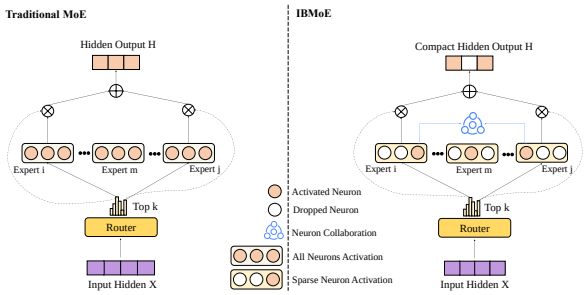


Figure 1: The differences between classical MoE and IBMoE. MoE focuses on expert-level specialization, while IBMoE enhances fine-grained, neuron-level specialization within each expert through sparse neuron activation and neuron collaboration.

Beyond PEFT, mixture-of-experts (MoE) offers a promising conditional computation paradigm for balancing capacity and efficiency. Instead of activating all parameters for every input, MoE routes tokens to a subset of specialized experts, enabling scalable models with constant per-token computation (Bengio et al., 2015; Lepikhin et al., 2020; Zadouri et al., 2023; Gao et al., 2024; Zhu et al., 2025). Recent efforts have combined MoE with PEFT (Li et al., 2025), MoCE (Gao et al., 2025) introduces a dual-level MoE framework for causal intervention, and HMoE (Lin et al., 2025) uses hierarchical expert routing for code search. However, such methods often encounter challenges related to knowledge redundancy and parameter inefficiency. In addition, (He et al., 2024) point out that MoE architectures exhibit more pronounced redundancy than dense models. Consequently, addressing knowledge redundancy in MoE has become a

critical challenge, and prior research has generally pursued two main lines of approach.

On the one hand, a line of work focuses on encouraging expert specialization. Some approaches design task-aware routing strategies to guide expert selection (Liu et al., 2024; Zhang et al., 2024). Others partition dense models into experts or construct parameter-isolated experts to improve specialization (Zuo et al., 2022; Zhu et al., 2024b). However, these methods mainly focus on improving specialization at the expert level, while redundancy within each expert remains largely unexplored.

On the other hand, enabling all experts to collaborate efficiently remains an important challenge (Cai et al., 2024). Most methods rely on routing mechanisms to assign tokens to multiple experts, often combined with balance constraints to stabilize utilization (Zhu et al., 2023). Some recent studies further reduce representational similarity across experts to mitigate collapse (Dai et al., 2024; Do et al., 2025). However, these methods still operate at the level of expert routing, without explicitly modeling interactions within experts or the underlying differences in their internal representations. As a result, the notion of expert knowledge diversity remains implicit and under-constrained.

To address these limitations, we propose IBMoE, a parameter-efficient MoE framework inspired by the Information Bottleneck principle (Tishby et al., 2000; Alemi et al., 2016). IBMoE aims to retain only task-relevant information while reducing redundant neuron activations. Specifically, each expert is equipped with learnable queries that induce sparse neuron activation under an IB constraint, enabling fine-grained control over activated neurons. In addition, we introduce a neuron-level collaboration module that explicitly encourages complementary specialization across experts, improving knowledge diversity and coordination. As shown in Figure 1, IBMoE shifts the focus from expert-level routing to neuron-level sparse activation and collaboration, enabling more efficient utilization of model capacity. The major contributions of our work can be summarized as follows:

- We propose IBMoE, an information bottleneck-based MoE framework that reduces neuron-level redundancy and enables fine-grained expert collaboration.
- We introduce a sparse neuron activation mechanism driven by learnable queries to selectively retain task-relevant information. Moreover, a neuron collaboration module is introduced to explicitly guide all experts to cooperate at the neuron-level with higher efficiency.
- We conduct extensive experiments to validate the effectiveness and superiority of our method. The experiments indicate that IBMoE achieves the state-of-the-art performance across various datasets, while reducing the inference time by 63% and the memory consumption by 48.5%.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) aims to adapt large language models to downstream tasks while updating only a small number of parameters. Existing PEFT methods can generally be divided into prompt-based and adapter-based approaches. Prompt-based methods optimize learnable prompts or virtual tokens while keeping the backbone frozen. Representative methods include Prompt-Tuning (Lester et al., 2021), P-Tuning (Liu et al., 2021), and Prefix-Tuning (Li and Liang, 2021). BitFit (Zaken et al., 2021) further reduces trainable parameters by updating only bias terms. Adapter-based methods instead introduce trainable modules into the backbone architecture. IA3 (Liu et al., 2022) scales intermediate activations using learnable vectors, while LoRA (Hu et al., 2021) performs low-rank parameter updates through decomposed matrices. Subsequent works further improve efficiency and expressiveness through alternative adapter designs (Braga et al., 2024; Ren et al., 2024). Our method follows the PEFT paradigm and combines it with sparse MoE modeling for efficient adaptation.

2.2 Mixture of Experts

Mixture-of-Experts (MoE) enables conditional computation by routing inputs to a subset of specialized experts. Existing MoE studies mainly focus on three directions. The first direction scales model capacity while maintaining constant per-token computation. Representative approaches such as GShard (Lepikhin et al., 2020), Switch Transformer (Fedus et al., 2022), and DeepSeek-MoE (Dai et al., 2024) improve scalability through sparse expert routing and fine-grained expert designs. The second direction reduces computational cost through knowledge separation or expert decomposition. For example, Llama-MoE (Zhu et al.,

2024b) and MoeBERT (Zuo et al., 2022) partition dense model parameters into multiple experts to improve efficiency. The third direction integrates MoE with PEFT adapters. MoLoRA (Zadouri et al., 2023) decomposes LoRA into multiple experts, while MoLA (Gao et al., 2024), MoR1E (Liu et al., 2024), and HMoRA (Liao et al., 2025) further improve expert routing and specialization. However, existing methods mainly focus on expert-level routing and specialization, while neuron-level redundancy and explicit modeling of expert diversity remain underexplored.

2.3 Information Bottleneck

Information Bottleneck (IB) aims to compress representations while preserving task-relevant information (Tishby et al., 2000). Prior studies have shown that IB-based representation learning can improve generalization by suppressing irrelevant information in hidden representations (Kawaguchi et al., 2023). To improve the optimization of conventional IB objectives in deep neural networks, several variants such as DeepIB (Tishby and Zaslavsky, 2015) and DIB (Strouse and Schwab, 2017) have been proposed. IB has been successfully applied to graph learning (Yu et al., 2021; Zhang et al., 2025), adversarial training (Chen et al., 2021), model compression (Lorenzen et al., 2021), and representation compression (Sheng et al., 2022). More recently, Zhu et al. (2024a) introduced IB to suppress noisy retrieved information in LLM-based generation. Different from prior IB applications, our method incorporates IB into MoE-based PEFT to reduce neuron-level redundancy and improve expert specialization.

3 Preliminaries

Mixture of Experts: In general, the MoE architecture consists of N feed-forward neural networks, denoted as $\{E\}_{i=1}^{N}$, which process various inputs through a gating mechanism. Specifically, the gating mechanism is a function equipped with the trainable parameters $W_r \in \mathbb{R}^{d_{in} \times N}$. For each input $x \in \mathbb{R}^{d_{in} \times 1}$, the gate function maps it to an N-dimensional discrete probability vector, which represents the contribution of each expert. Then, a routing algorithm, such as the top-k or soft routing method, is used to allocate the input to the experts. In this paper, we select the top-k experts with the highest contributions to process the input. The output h_i at i-th token is calculated as:

$$G_i = \text{TopK}(\text{softmax}(W_r^\top x_i) \tag{1}$$

$$h_i = \sum_{e=1}^{N} G_{i,e} E_e(x_i) \tag{2}$$

where $G_{i,e}$ represents the contribution of e-th expert at i-th token.

Low-Rank Adaptation: Inspired by the natural low intrinsic dimension, LoRA introduces low-rank decomposition for optimizing the weight matrix of pre-trained LLMs to reduce the number of training parameters. Let the weight matrix of a pre-trained linear layer be $W_0 \in \mathbb{R}^{d_{in} \times d_{out}}$. LoRA learns two low-rank matrices $A \in \mathbb{R}^{r \times d_{in}}$ and $B \in \mathbb{R}^{d_{out} \times r}$ to update W_0, where $r \ll \min(d_{in}, d_{out})$. During training, W_0 is frozen and only matrices A and B are updated. Given an input x_i, the output of the LoRA layer can be expressed as:

$$\begin{aligned} h_i &= W_0^\top x_i + \frac{\alpha}{r} BAx_i \\ &= W_0^\top x_i + \Delta W x_i \end{aligned} \tag{3}$$

where α is a scale hyper-parameter.

MoE and LoRA: Some studies integrate the advantages of LoRA and MoE by constructing multiple LoRA adapters as experts, known as LoRA-MoE, to achieve expert specialization on different tasks. Combining Equations 2 and 3, we have:

$$h_i = W_0^\top x_i + \sum_{e=1}^{N} G_{i,e} \Delta W_e x_i \tag{4}$$

where ΔW_e is the weight matrix of e-th expert.

4 Methodology

In this section, we describe the details of the IB-MoE framework with Sparse Neuron Activation (SNA) and Neuron Collaboration (NC), as shown in Figure 2. Specifically, SNA integrates the IB principle into the MoE framework to capture each expert's knowledge characteristics. This integration reduces redundant information at the neuron level and enhances expert specialization. In turn, we introduce a NC mechanism that maximizes knowledge diversity across experts, encouraging collaboration and improving parameter utilization.

4.1 Sparse Neuron Activation

Typically, the knowledge characteristics of experts are implicitly determined by the routing mechanism, which makes it difficult to analyze and control redundant knowledge. Motivated by the IB

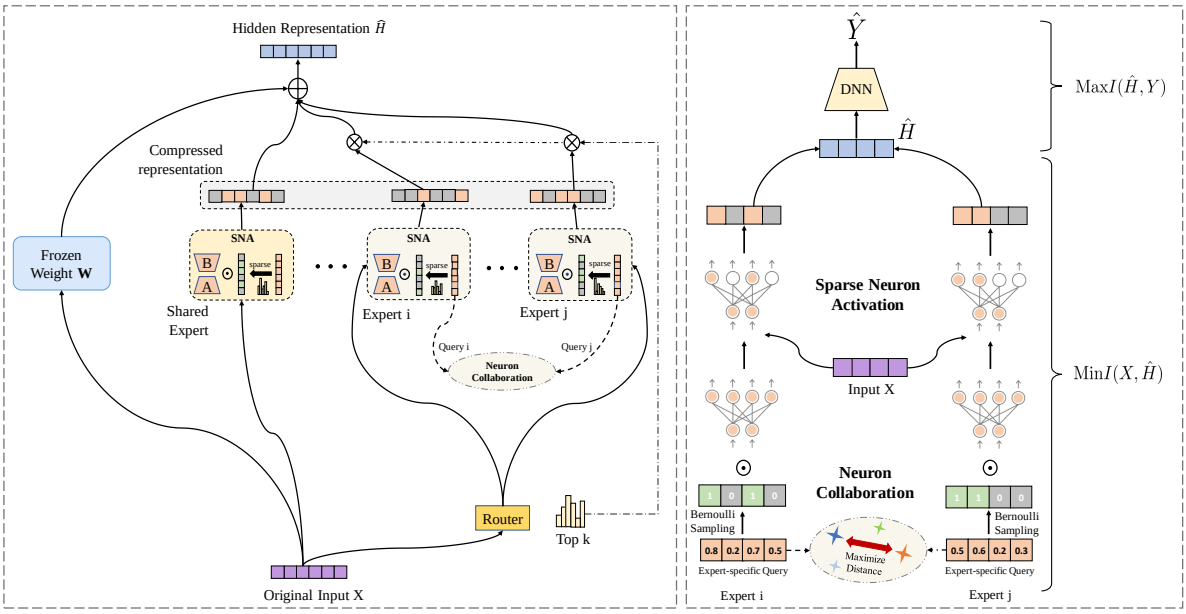


Figure 2: The overall framework of IBMoE consists of two components: Sparse Neuron Activation (SNA) and Neuron Collaboration (NC). SNA utilizes the IB to learn a neuron-level query vector for discarding irrelevant details. The NC strategy is then introduced to reduce redundant knowledge by modeling interactions among experts.

theory, we introduce the SNA module, which explicitly filters out task-irrelevant signals and preserves task-specific information for each expert. At a finer granularity, not all neurons within an expert contribute equally to a task. Therefore, SNA activates only the most relevant neurons, analogous to sparse expert activation at the expert level. In this way, SNA captures task-specific characteristics by modeling differences in neuron activation distributions across experts and enhances fine-grained specialization. Specifically, let the inputs and labels be denoted as X and Y, respectively, and let $\hat{H}$ represent the compressed representation of X. Each expert is trained to minimize the mutual information $I(X;\hat{H})$ while maximizing $I(\hat{H};Y)$, so as to discard irrelevant information and preserve task-relevant knowledge:

$$\begin{aligned} L_{ib} &= \beta I(X,\hat{H}) - I(\hat{H},Y) \\ &= \mathop{\mathbb{E}}_{(x,\hat{h})\sim p(x,\hat{h})} \beta \log \frac{p(\hat{h}|x)}{p(\hat{h})} \\ &\quad - \mathop{\mathbb{E}}_{(\hat{h},y)\sim p(\hat{h},y)} \log \frac{p(y|\hat{h})}{p(y)} \end{aligned} \tag{5}$$

where β is a hyper-parameter that used to controls the compression of X.

However, computing $p(\hat{h})$ and $p(y|\hat{h})$ directly is intractable. To solve this, with the concept of Variational Information Bottleneck (VIB) (Alemi et al., 2016), we use two variational distributions $r(\hat{h})$ and $q(y|\hat{h})$ to approximate $p(\hat{h})$ and $p(y|\hat{h})$, which allows us to optimize the upper bound of L_{ib}. For the former term in Equation 5, we have:

$$\begin{aligned} I(X;\hat{H}) &= \mathop{\mathbb{E}}_{(x,\hat{h})\sim p(x,\hat{h})} \log \frac{p(\hat{h}|x)}{r(\hat{h})} - \mathop{\mathbb{E}}_{\hat{h}\sim p(\hat{h})} \log \frac{p(\hat{h})}{r(\hat{h})} \\ &= \mathop{\mathbb{E}}_{x\sim p(x)} \mathrm{KL}\left(p(\hat{h}|x)||r(\hat{h})\right) - \mathrm{KL}\left(p(\hat{h})||r(\hat{h})\right) \end{aligned} \tag{6}$$

By non-negativity of KL divergence, $\mathrm{KL}(p(\hat{h}) \,\|\, r(\hat{h})) \geq 0$, hence

$$I(X;\hat{H}) \;\leq\; \mathbb{E}_{x\sim p(x)} \mathrm{KL}\Big(p(\hat{h} \mid x) \,\|\, r(\hat{h})\Big) \tag{7}$$

Similarly, for the latter term in Equation 5 and based on the fact that $\mathrm{KL}(p(y|\hat{h})||q(y|\hat{h})) \geq 0$, the lower bound is calculated as:

$$\begin{aligned} \mathop{\mathbb{E}}_{(\hat{h},y)\sim p(\hat{h},y)} \log \frac{p(y|\hat{h})}{p(y)} &\geq \mathop{\mathbb{E}}_{(\hat{h},y)\sim p(\hat{h},y)} \log \frac{q(y|\hat{h})}{p(y)} \\ I(\hat{H};Y) &\geq \mathop{\mathbb{E}}_{(\hat{h},y)\sim p(\hat{h},y)} \log q(y|\hat{h}) + \mathrm{H}(Y) \end{aligned} \tag{8}$$

where $\mathrm{H}(Y)$ is the entropy of Y, which is independent of the optimization and can be ignored.

Then, for optimizing Equation 7, we first add a task-specific query $V_i \in \mathbb{R}^{d_{out}}$ for E_i to represent the importance of the neurons in ΔW_i. Before the forward process, E_i uses V_i to generate a binary mask $M_i = [m_1, m_2, ..., m_{d_{out}}]^\top \in \{0,1\}^{d_{out}}$ via

a Bernoulli-based importance sampling, where M_i indicates which neurons should be dropped. M_i is generated as follows:

$$M_i = \mathrm{Ber}\left(\frac{V_i - \min(V_i) + \epsilon}{\max(V_i) - \min(V_i) + \epsilon}\right) = \mathrm{Ber}(\hat{V}) \tag{9}$$

where $\mathrm{Ber}(.)$ is a function used for Bernoulli sampling, and ϵ is small positive number. Note that the ϵ in the denominator prevents division by zero erros, while ϵ in the numerator ensures that each neuron becomes activated when all neurons have equal importance. The compressed hidden representation can be expressed as:

$$\hat{h_i} = W_0^\top x_i + \sum_{e=1}^{N} \frac{d_{out} G_{i,e} \Delta \hat{W}_e x_i}{\mathrm{sum}(M_e)} \tag{10}$$

where $\Delta \hat{W}_e = M_e \circ \Delta W_e$, $\circ$ denotes element-wise multiplication at the column level, and $\mathrm{sum}(.)$ is a summation function. We provide the detailed implementation and annotations in Appendix A.

In this paper, Bernoulli sampling is used as the prior distribution, denoted as $r(\hat{h}) = \mathrm{Ber}(\pi)$. We aim for the expected number of activated neurons to follow a Bernoulli distribution with a probability of π, which enables controllable compression of invalid information. Compared with h, the computation of $\hat{h}$ additionally depends on the query V of each expert, where V is independent of x. Consequently, we have $p(\hat{h}|x) = p(M|V)$ and $r(\hat{h}) = r(M)$. Therefore, a sparsity constraint is designed to minimize $\mathrm{KL}\left(p(\hat{h}|x)||r(\hat{h})\right)$:

$$L_s = \sum_{i=1}^{N} \left[\bar{V_i} \cdot \log \frac{\bar{V_i}}{\pi} + (1 - \bar{V_i}) \cdot \log \frac{1 - \bar{V_i}}{1 - \pi}\right] \tag{11}$$

where $\bar{V_i}$ represents the mean value of $\hat{V_i}$. By optimizing V with L_s, it encourages each expert activates an expected proportion π of neurons with heterogeneous importance.

In practical, considering the non-differentiability of Bernoulli sampling during training, we use Gumbel-Softmax reparameterization trick to approximate the Bernoulli distribution (Jang et al., 2017). The sampling process for $\hat{V_i}$ is computed as follows:

$$M_i = \sigma\left(\frac{\log \hat{V_i} + g}{\tau}\right) \tag{12}$$

where $g = -\log(-\log(u)), u \sim U(0,1)$, $\tau = 0.5$ is a hyper-parameter for controlling sampling randomization, and $\sigma(.)$ is sigmoid function.

Finally, for Equation 8, $q(y|\hat{h})$ is typically achieved by a network $q_\theta(\hat{h})$ used for the next token prediction task, with the loss function being cross-entropy. The loss function is defined as:

$$L_p = \sum_{x_i \in X} -y_i \log(q_\theta(p_\phi(x_i))) \tag{13}$$

where $p_\phi(.)$ represents a learnable neural network.

4.2 Neuron Collaboration

In this section, we introduce Neuron Collaboration (NC) to enhance the knowledge diversity among experts. Traditional MoE frameworks suffer from knowledge redundancy due to task similarity, but in IBMoE, we tackle this by applying a feature-level diversity loss. It controls the knowledge differences among experts, promoting task specialization. Specifically, a query's distribution represents the knowledge characteristics of an expert. For two queries $\hat{V_i}$ and $\hat{V_j}$, we minimize their similarity, encouraging experts to focus on different feature dimensions through diverse query distributions. The diversity constraint is expressed as:

$$L_d = \sum_{i,j}^{N} \mathrm{cosine}(\hat{V_i}, \hat{V_j}) \tag{14}$$

where $\mathrm{cosine}(.,.)$ is a function for computing the cosine similarity between two vectors.

We follow the settings of vanilla MoE frameworks that introducing an auxiliary loss for the router to avoid expert collapse. Let P_r represent the probability distribution of tokens assigned to the experts. The objective function is defined as:

$$L_b = \frac{N}{k} \sum_{i=1}^{N} \frac{c_i}{T} * \mathrm{mean}_e(P_r) \tag{15}$$

where k is the number of activated expert, $\mathrm{mean}_e(.)$ represents the expert-wise mean value, c_i is the number of tokens allocated to i-th expert and T represents the total number of tokens.

Combining Equations 11, 13, 14 and 15, the finally loss function is expressed as:

$$L = L_p + \beta L_s + \gamma L_d + \lambda L_b \tag{16}$$

The complete pseudo-code is provided in Appendix A.3.

5 Experimental Setup

5.1 Datasets

We evaluate IBMoE on seven NLP datasets, including three GLUE tasks, three question answering (QA) datasets, and one instruction-following dataset from SuperNI. For GLUE, we select one representative task from each category: **CoLA** (Warstadt et al., 2019), **MRPC** (Dolan and Brockett, 2005), and **RTE** (Wang et al., 2019). For QA evaluation, we use **ScienceQA (SQA)** (Lu et al., 2022), **OpenbookQA (OBQA)** (Mihaylov et al., 2018), and **CommonSenseQA (CSQA)** (Talmor et al., 2019). For instruction-following evaluation, we include **Convai3** from SuperNI (Wang et al., 2022). Detailed dataset formatting and preprocessing procedures are provided in Appendix B.

5.2 Base Models and Baselines

We evaluate IBMoE on three base models of different scales: LLaMA-3.2-3B (L-3B) (Dubey et al., 2024), and Mistral-7B-v0.1 (M-7B) (Jiang et al., 2023). To evaluate the effectiveness of our model, we compare our IBMoE with recent parameter-efficient fine-tuning (PEFT)-based and MoE-based methods. For PEFT-based methods, we select **LoRA** (Hu et al., 2021). For MoE-based methods, we employ **MoLA** (Gao et al., 2024), **AlphaMoLA** (Qing et al., 2024), **AdaMoLE** (Liu and Luo, 2024) and **HMoRA** (Liao et al., 2025). Detailed descriptions of baseline methods are provided in Appendix C.

5.3 Evaluation Metrics

We evaluate IBMoE from four perspectives: task performance, parameter efficiency, training cost, and inference efficiency. For GLUE and QA tasks, we report **accuracy**, and for generation tasks, we use **rougeL** to measure textual similarity. To evaluate efficiency, we report the number of activated parameters (**Act Param**), training time (**GPU-hour**), and inference cost **Time** and **Peak Memory**. Overall, these metrics provide a comprehensive evaluation of model effectiveness and efficiency.

5.4 Implementation Details

The hyper-parameters are set as follows: (1) **Training Parameters**: We use the Adafactor optimizer with a learning rate of $2e^{-4}$, a batch size of 128. The number of epochs is set to 10. (2) **LoRA adapter**: We adopt the default LoRA configuration, with the intrinsic LoRA rank of experts set to 8 for all the baselines. (3) **IBMoE Parameters**: We apply 5 LoRA experts to every other MLP layer, with one shared expert handling all inputs. During forward processing, the top-k is set to 2, selecting one specific expert and the shared expert for each token. We set $\pi = 0.6$, $\beta = 0.1$, $\gamma = 0.1$, and $\lambda = 0.01$. All experiments are conducted on the NVIDIA A800 GPU.

6 Experimental Results

6.1 Main Results

Table 1 compares IBMoE with representative PEFT-based and MoE-based baselines. IBMoE achieves the best overall performance on all three base models while using the fewest activated parameters. Compared with sparse baselines such as SparseAdapter and MoLoRA, IBMoE consistently delivers stronger accuracy with substantially lower parameter usage, indicating that dynamic neuron-level sparsity is more effective than static adapter pruning or expert-level activation alone. These results suggest that modeling fine-grained neuron importance within experts is crucial for reducing redundancy and improving parameter efficiency.

6.2 Ablation Studies

In Table 2, we conduct ablation studies to evaluate the effectiveness of IBMoE's components. First, we experiment without L_b. We find that imbalanced routing leads to some experts handling disproportionate amounts of data by activating more neurons, which increases the **Act Param**. Next, we remove the sparsity constraint, which makes the experts fail to focus on the most relevant information, and unstable neuron sampling further reduces performance. Finally, we remove the diversity loss, which worsens IBMoE's performance. Even with a shared expert, omitting L_d can still lead to knowledge redundancy among the experts. Additionally, without any loss constraints, the number of activated parameters in IBMoE approximately doubles because the inactive SNA module causes the model to activate all neurons to complete the tasks. The complete results are provided in Appendix D.1.

6.3 Further Analyses

We conduct additional analyses to better understand the effectiveness and efficiency of IBMoE. More experimental results are deferred to the Appendix D.

Table 1: Comparison of performance and parameter efficiency with other SOTA methods across different model architectures. Average accuracy across seven datasets (**Avg**) is reported to evaluate overall performance. Best results are highlighted in **bold**.

Models	Methods	Act Param ↓	Datasets							Avg
			MRPC	CoLA	RTE	SQA	OBQA	CSQA	Convai	
L-3B	LoRA	7.6M	83.24	**86.86**	88.44	92.21	79.40	77.73	30.32	76.88
	SparseAdapter	3.04M	84.86	85.71	87.73	93.12	80.20	76.49	30.84	76.99
	MoLA	23.59M	**85.77**	82.13	84.12	91.29	80.00	77.16	28.49	75.57
	AlphaLoRA	26.16M	84.90	82.13	83.39	92.51	80.00	**78.63**	29.35	75.84
	AdaMoLE	dynamic	84.69	84.27	87.73	91.68	**80.40**	77.81	29.56	76.59
	HMoRA	30.91M	83.88	86.58	84.47	92.53	80.20	77.31	31.36	76.62
	IBMoE	**3.01M**	84.69	86.10	**88.80**	**93.16**	80.20	78.13	**31.92**	**77.57**
M-7B	LoRA	20.50M	85.86	86.23	90.97	94.86	86.20	79.85	**31.88**	79.40
	SparseAdapter	8.20M	87.07	85.33	91.33	95.05	88.80	81.16	27.35	79.44
	MoLA	43.40M	86.95	87.44	88.80	95.14	88.20	83.37	29.92	79.97
	AlphaLoRA	43.40M	87.13	**87.91**	91.70	95.00	89.20	**84.00**	29.59	80.65
	AdaMoLE	dynamic	85.91	86.10	92.05	94.90	87.80	82.55	**31.88**	80.17
	HMoRA	44.20M	87.48	87.53	90.28	94.60	88.60	82.64	31.07	80.31
	IBMoE	**8.22M**	**88.58**	87.53	**92.77**	**95.68**	**90.40**	83.05	31.83	**81.40**

Model	RTE	
	Acc (%↑)	Act Params ↓
IBMoE	88.43	7.89M
$-L_b$	87.95	7.91M
$-L_b - L_s$	87.06	7.07M
$-L_b - L_s - L_d$	86.29	13.16M

Table 2: Ablation results of IBMoE on the RTE dataset.

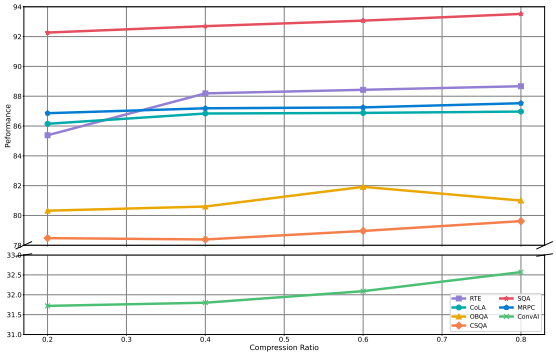


Figure 3: Compression Ratio π with different values.

Analysis of Compression Ratio. In IBMoE, the parameter π controls the proportion of activated neurons. The results in Figure 3 illustrate the model's performance under various compression ratios across multiple datasets. We observe that model performance generally improves as more neurons are activated. However, this improvement gradually diminishes and may eventually reverse as π continues to increase. This observation suggests the presence of neuron redundancy in conventional MoE methods, which can potentially degrade model performance by activating task-irrelevant neurons. Moreover, the model's performance on the RTE dataset drops by approximately 3% when π is set to 0.2, indicating that an excessively low activation ratio may limit the model's capacity to capture task-relevant information.

Analysis of Hyper-parameters. The IBMoE framework introduces three hyper-parameters (λ, β, and γ) for the components of the loss function. To analyze the sensitivity of these hyper-parameters, we conduct experiments by varying their values. Specifically, we adjust one parameter (e.g., λ) while fixing the others to their default values. The same procedure is repeated for β and γ. As shown in Figure 4, the proposed method exhibits strong robustness to hyper-parameter variations. Notably, setting lower values (e.g., 0.01 or 0.001) leads to a slight improvement in model performance. However, when λ is set to 0.1, the scaling of the balance loss L_b dominates the primary task loss L_p. In this case, tokens with similar semantic information are forcibly assigned to different experts to ensure load balancing, which makes it difficult for experts to process semantically diverse tokens under sparse representations.

Table 3: Training and Inference efficiency of IBMoE.

Model	**Inference with 1,000 Samples**		**Training on the RTE Dataset**	
	Time (s) ↓	Peak Memory (MB) ↓	Train Param (%)↓	GPU-hour ↓
LoRA	60	116	0.3	0.77h
SparseAdapter	67.34	232	0.3	0.81h
MoLA	739.3	1452.6	1.51	2.93h
AlphaLoRA	285.7	812	1.51	2.58h
AdaMoLE	329.8	146.2	0.53	1.97h
HMoRA	295.6	842.1	3.15	3.55h
IBMoE	109.3	75.7	0.47	0.88h

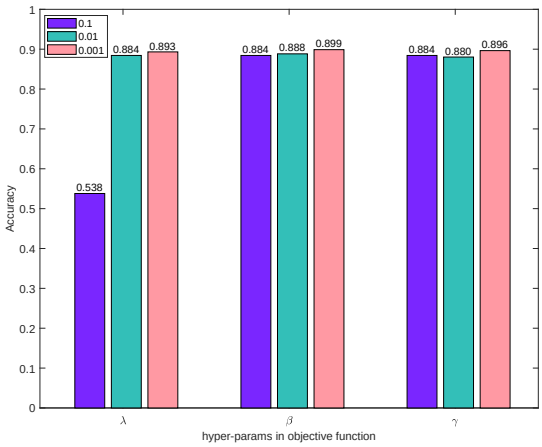


Figure 4: Sensitivity of hyper-parameters.

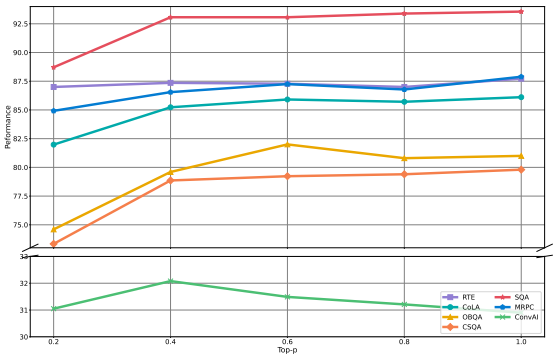


Figure 5: Impact of p on the different tasks.

Analysis of Greedy Sampling. To further analyze neuron activation, we replace Bernoulli sampling with greedy sampling during inference. Specifically, top-p sampling is applied on V to activate the most relevant neurons, where the binary mask is defined as $M_i = \mathrm{TopP}(\hat{V})$. Figure 5 presents the results of IBMoE with greedy sampling under different activation ratios. We observe that greedy sampling performs worse than Bernoulli-based importance sampling under the same activation scale, since it focuses only on highly ranked neurons and lacks global feature exploration. As p increases, its performance gradually approaches that of importance sampling, indicating that importance sampling achieves a better trade-off between performance and efficiency.

Analysis of the Inference Efficiency. We also compare the inference performance of IBMoE with other methods to evaluate its efficiency. To ensure the fairness and reproducibility of all methods, we conduct inference using a single NVIDIA A800 GPU with 80 GB memory. Specifically, we evaluate Mistral-7B model in 16-bit precision on 1,000 randomly sampled test examples, measuring inference latency and peak memory usage of the adapters with a batch size of 16. In Table 3, our proposed method achieves the best performance in both time and memory efficiency, with improvements of 63% in time and 48.5% in memory efficiency compared to the SOTA MoE-based PEFT methods. We attribute this advantage to the fact that the trained IBMoE significantly reduces computational cost and memory consumption by compressing the weight matrices of LoRA experts.

7 Conclusions and Future Work

In this paper, we propose a novel parameter-efficient MoE method that introduces an IB framework and multiple constraints to achieve fine-grained reduction of neuronal redundancy, fostering closer collaboration among experts. Extensive experiments show that our method achieves superior performance comparable to state-of-the-art methods, and reduces inference time by 63% and memory consumption by 48.5%. Furthermore, ablation studies validate the effectiveness of each IBMoE component. In future work, we will explore applications in continual learning by capitalizing on the knowledge specialization inherent in MoE.

Limitations

Despite its effectiveness, IBMoE has several limitations. First, although we adopt a top-k sparse routing strategy for efficiency and specialization, sparse routing introduces additional overhead for token allocation and expert dispatch in distributed settings. In contrast, soft routing avoids this issue but incurs higher computation during expert activation. Second, IBMoE introduces additional hyperparameters and structural components compared with standard PEFT methods, such as the number of experts and activation settings, which may require manual tuning when adapting to new tasks or domains. Finally, while we evaluate IBMoE on a wide range of NLP tasks, its behavior under continual learning settings remains unexplored.

References

Tanay Aggarwal, Angelo Salatino, Francesco Osborne, and Enrico Motta. 2026. Large language models for scholarly ontology generation: An extensive analysis in the engineering field. *Information Processing & Management*, 63:104262.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, arXiv:1612.00410.

Zijian Bai, Yucheng Zheng, Peng Yang, Siyang Liu, Yiyuan Zhang, and Yuanyuan Chang. 2025. Dcrl-rec: Dual-domain contrastive reinforcement large language model for recommendation. *Information Processing & Management*, 62:104140.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2015. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, arXiv:1511.06297.

Marco Braga, Alessandro Raganato, Gabriella Pasi, and 1 others. 2024. Adakron: An adapter-based parameter efficient model tuning with kronecker product. In *2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC-COLING 2024-Main Conference Proceedings*, pages 350–357.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2024. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*, arXiv:2407.06204.

Jiawei Chen, Ziqi Zhang, Xinpeng Xie, Yuexiang Li, Tao Xu, Kai Ma, and Yefeng Zheng. 2021. Beyond mutual information: Generative adversarial network for domain adaptation using information bottleneck constraint. *IEEE Transactions on Medical Imaging*, 41(3):595–607.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, arXiv:2401.06066.

Giang Do, Hung Le, and Truyen Tran. 2025. Simsmoe: Toward efficient training mixture of experts via solving representational collapse. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2012–2025.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*, pages 9–16.

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

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The International Journal of Man-Machine Studies*, 23(120):1–39.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*, arXiv:2402.08562.

Yifei Gao, Ning Xu, Hongshuo Tian, Bolun Zheng, Jinbo Cao, and An-An Liu. 2025. Mixture of causal experts: A causal perspective to build dual-level mixture-of-experts models. *Expert Systems with Applications*, 280:127422.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, arXiv:2501.12948.

Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. 2022. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2184–2190.

Shwai He, Daize Dong, Liang Ding, and Ang Li. 2024. Towards efficient mixture of experts: A holistic study of compression techniques. *arXiv preprint arXiv:2406.02500*, arXiv:2406.02500.

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. 2023. Merging experts into one: Improving computational efficiency of mixture of experts. *arXiv preprint arXiv:2310.09832*, arXiv:2310.09832.

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*, arXiv:2106.09685.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of 5th International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*, arXiv:2310.06825.

Kenji Kawaguchi, Zhun Deng, Xu Ji, and Jiaoyang Huang. 2023. How does information bottleneck help deep learning? In *Proceedings of the 40th International Conference on Machine Learning*, pages 16049–16096.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, arXiv:2006.16668.

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

Linjie Li, Zhenyu Wu, and Yang Ji. 2025. Mote: Mixture of task-specific experts for pre-trained model-based class-incremental learning. *Knowledge-Based Systems*, page 113795.

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

Mengqi Liao, Wei Chen, Junfeng Shen, Shengnan Guo, and Huaiyu Wan. 2025. Hmora: Making llms more effective with hierarchical mixture of lora experts. In *The Thirteenth International Conference on Learning Representations*.

Zexu Lin, Xu Zhang, Xiaoyu Hu, Pengfei Li, Deyu Zhou, and Darong Lai. 2025. Hierarchical mixture-of-experts model for unified code search. *Information Fusion*, page 103489.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35(120):1950–1965.

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*, arXiv:2110.07602.

Yijiang Liu, Rongyu Zhang, Huanrui Yang, Kurt Keutzer, Yuan Du, Li Du, and Shanghang Zhang. 2024. Intuition-aware mixture-of-rank-1-experts for parameter efficient finetuning. *arXiv preprint arXiv:2404.08985*, arXiv:2404.08985.

Zefang Liu and Jiahua Luo. 2024. Adamole: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. *arXiv preprint arXiv:2405.00361*, arXiv:2405.00361.

Stephan Sloth Lorenzen, Christian Igel, and Mads Nielsen. 2021. Information bottleneck: Exact analysis of (quantized) neural networks. *arXiv preprint arXiv:2106.12912*, arXiv:2106.12912.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, arXiv:2303.08774.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. 2024. Alphalora: Assigning lora experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20511–20523.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Melora: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3064.

Changchong Sheng, Li Liu, Wanxia Deng, Liang Bai, Zhong Liu, Songyang Lao, Gangyao Kuang, and Matti Pietikäinen. 2022. Importance-aware information bottleneck learning paradigm for lip reading. *IEEE Transactions on Multimedia*, 25:6563–6574.

DJ Strouse and David J Schwab. 2017. The deterministic information bottleneck. *Neural computation*, 6(120):1611–1630.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4149–4158.

Shaopeng Tang, Lin Li, Xiaohui Tao, Leqi Zhong, and Qing Xie. 2025. MATO: A model-agnostic training optimization for aspect sentiment triplet extraction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1648–1662.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*, physics/0004057.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, arXiv:2302.13971.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of 7th International Conference on Learning Representations*.

Yifan Wang, Yixin Cao, Zeping Li, Bowen Dong, Guangnan Ye, and Hongfeng Chai. 2025. Multi-hgpt: Multi-task heterogeneous graph prompt tuning. *Information Processing & Management*, 62:104236.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, and 1 others. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 5085–5109.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. 2021. Recognizing predictive substructures with subgraph information bottleneck. *IEEE transactions on pattern analysis and machine intelligence*, 46(3):1650–1663.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, arXiv:2309.05444.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, arXiv:2106.10199.

Rongyu Zhang, Yulin Luo, Jiaming Liu, Huanrui Yang, Zhen Dong, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, Yuan Du, and 1 others. 2024. Efficient Deweahter Mixture-of-Experts with Uncertainty-Aware Feature-Wise Linear Modulation. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pages 16812–16820.

Shuai Zhang, Junfeng Fang, Xuqiang Li, ALAN XIA, Ye Wei, Wenjie Du, Yang Wang, and 1 others. 2025. Iterative substructure extraction for molecular relational learning with interactive graph information bottleneck. In *The Thirteenth International Conference on Learning Representations*.

Junda Zhao, Yuliang Song, and Eldan Cohen. 2025. Variational prefix tuning for diverse and accurate code summarization using pre-trained language models. *Journal of Systems and Software*, page 112493.

Kun Zhu, Xiaocheng Feng, Xiyuan Du, Yuxuan Gu, Weijiang Yu, Haotian Wang, Qianglong Chen, Zheng Chu, Jingchang Chen, and Bing Qin. 2024a. An information bottleneck perspective for effective noise filtering on retrieval-augmented generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1069.

Shaolin Zhu, Leiyu Pan, Dong Jian, and Deyi Xiong. 2025. Overcoming language barriers via machine translation with sparse mixture-of-experts fusion of large language models. *Information Processing & Management*, 62:104078.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024b. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*, arXiv:2406.16554.

Yun Zhu, Nevan Wichers, Chu-Cheng Lin, Xinyi Wang, Tianlong Chen, Lei Shu, Han Lu, Canoee Liu, Liangchen Luo, Jindong Chen, and 1 others. 2023. Sira: Sparse mixture of low rank adaptation. *arXiv preprint arXiv:2311.09179*, arXiv:2311.09179.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2022. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675*, arXiv:2204.07675.

A Notation and Explanation

A.1 Notations of IBMoE

In Table 4, we summarize the notations used in IBMoE and explicitly specify their tensor dimensions. On the other hand, in Equation 10, the column-level masking operation:

$$\Delta\hat{W}_e = M_e \circ \Delta W_e \tag{17}$$

is defined as follows. Let $M_e \in \{0,1\}^{d_{\text{out}}}$ be a binary mask vector and $\Delta W_e \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the LoRA weight of expert e. The operation $\circ$ applies M_e to each column of ΔW_e:

$$(\Delta\hat{W}_e)_{j,:} = M_{e,j} \cdot (\Delta W_e)_{j,:}, \tag{18}$$

where $(\cdot)_{j,:}$ denotes the j-th row. This operation selectively activates neurons while preserving a complete computation graph during training. Note that the operator $\circ$ in IBMoE has a different definition from the conventional Hadamard product. However, due to the broadcasting mechanism in Python, the involved tensors are first expanded to compatible shapes and then multiplied element-wise. Therefore, we still use $\circ$ to denote this type of operation in IBMoE.

A.2 Detailed Forward Mechanism of SNA

Figure 6 shows the forward mechanism of the SNA module. During training, the neuron-level mask M_e is applied to the weight update ΔW_e to ensure that the full computation graph is preserved for backpropagation. This design allows gradients to flow through all neurons while enforcing sparse activation at the neuron level. During inference, the mask M_e is directly used to select the activated neurons for computing the compressed representation $\hat{H}$, without constructing the full computation graph. This decoupling between training and inference enables efficient optimization while maintaining sparse neuron activation at test time.

A.3 Pseudo Code of IBMoE

We provide pseudo-code to describe the optimization procedure of IBMoE. Specifically, Algorithm 1 illustrates the sparse activation process of SNA during forward computation. Algorithm 2 further presents the forward pass of an IBMoE layer. Finally, Algorithm 3 details the computation of the overall objective function.

B Details of Dataset

We provide detailed descriptions of all datasets used in this work.

B.1 GLUE Benchmark

The General Language Understanding Evaluation (GLUE) benchmark consists of multiple NLP tasks, including single-sentence classification, sentence similarity, and natural language inference.

We select three representative tasks:

- **CoLA** (Warstadt et al., 2019): A grammatical acceptability classification task based on linguistic theory corpora.
- **MRPC** (Dolan and Brockett, 2005): A paraphrase identification dataset from news articles.
- **RTE** (Wang et al., 2019): A natural language inference task constructed from Wikipedia and news sources.

B.2 Question Answering Datasets

We evaluate reasoning and commonsense capabilities using:

- **ScienceQA (SQA)** (Lu et al., 2022): A science-based QA dataset originally containing multimodal samples. We retain only text-based questions.
- **OpenbookQA (OBQA)** (Mihaylov et al., 2018): QA dataset requiring elementary science knowledge reasoning.
- **CommonSenseQA (CSQA)** (Talmor et al., 2019): Multiple-choice QA requiring commonsense reasoning.

B.3 Instruction Dataset

Convai3 Sentence Generation from SuperNI (Wang et al., 2022) is a dialogue generation task requiring response generation conditioned on conversational context.

B.4 Examples of formatted datasets

In the experimental settings, all datasets are drawn from the GLUE, QA, and SuperNI benchmarks. To ensure reproducibility, we provide representative examples of the formatted datasets to illustrate the construction of our training inputs, as shown in Figure 7. For datasets from the GLUE and QA

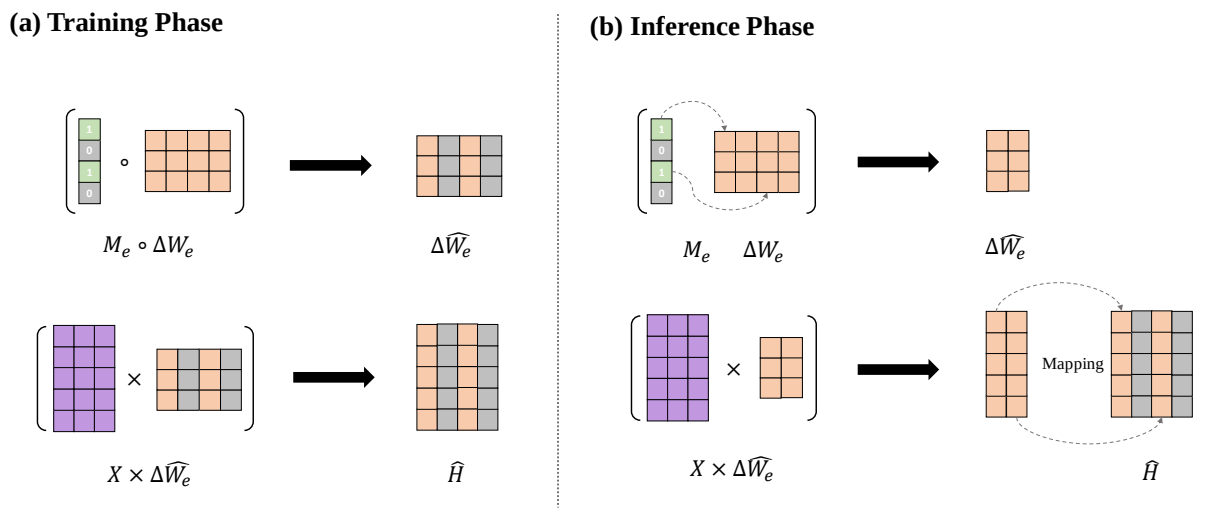


Figure 6: Forward mechanism of the SNA module. During the training stage, M_e is directly multiplied with ΔW_e to obtain sparse representations, which ensures that gradients can be backpropagated to the query mask. During the inference stage, the query mask is used to index the activated columns, enabling more efficient computation.

benchmarks, we adopt the Alpaca instruction format to construct the training inputs for all methods. For the ConvAI task, we follow the default instruction format provided by the dataset.

C Baseline Methods

We compare IBMoE with representative PEFT-based and MoE-based methods.

C.1 PEFT-based Methods

- **LoRA** (Hu et al., 2021): Introduces low-rank matrices for efficient adaptation.
- **SparseAdapter** (He et al., 2022): Applies structured sparsity to improve parameter efficiency.

C.2 MoE-based Methods

- **MoLA** (Gao et al., 2024): Allocates experts across layers to reduce redundancy.
- **AlphaLoRA** (Qing et al., 2024): Designs layer-wise expert allocation based on heavy-tailed regularization.
- **AdaMoLE** (Liu and Luo, 2024): Dynamically adjusts the number of active experts per layer.
- **HMoRA** (Liao et al., 2025): Combines token-level and task-level routing for finer-grained expert control.

Algorithm 1 Function `GSSampleMask`(V_i, τ, ϵ)

function GSSAMPLEMASK(V_i, τ, ϵ)
  $U \sim \mathcal{U}(0,1)$
  $\hat{V}_i \leftarrow (\min(V_i) - V_i + \epsilon)/(\max(V_i) - \min(V_i) + \epsilon)$
  $\hat{U} \leftarrow \log(\hat{V}_i) - \log(-\log(U + \epsilon) + \epsilon)$
  $S \leftarrow \sigma(\hat{U}/\tau)$
  $M \leftarrow \mathbf{0}^{|S|}$
  for $j = 1$ to $|S|$ **do**
    if $S[j] > 1 - \hat{V}_{i,j}$ **then**
      $M[j] \leftarrow 1$
    end if
  end for
  return M
end function

D Additional Results

D.1 Ablation Study Results

In table 5, We report the full ablation results across all datasets.

D.2 Different Expert Settings

In IBMoE, multiple constraints are introduced in the objective function. To investigate the effectiveness of these constraints under different numbers of experts and activated experts, we scale up each MoE layer and evaluate the performance of IBMoE. Specifically, we set the number of experts to 9, including 8 task-specific experts and 1 shared expert, with top-k set to 3. Experiments are conducted on three datasets from the GLUE

The formatted examples of QA datasets

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

Instruction:
Context: N/A
Question: As a source of light moves directly overhead of an object , what happens to it‘s shadow?
Options: (A) the object will get brighter
(B) the shadow will start to spin
(C) the size will decrease
(D) the light will lean to one side

Response:
Answer: The answer is C.

(a)

The formatted examples of Convai dataset

Definition: You are given a concatenated string consisting of two parts that are separated by a newline character. The first part is the user's initial question during a human computer conversation, the second part is a clarifying question asked by the computer in response to the question asked by the user. You should read the input, then generate a valid prediction of the user's response to the computer's clarifying question. A valid prediction will clarify the user's correct intent in the initial question.

Now complete the following example -
Input: User: Find information on raised garden beds and boxes.
Computer: do you need information on materials needed.
Output: no i am looking for pictures of raised garden beds not the materials needed.

(b)

Figure 7: The examples of training data.

Table 4: Summary of notations and tensor dimensions.

Symbol	Description	Dimension
x_i	Input of token i	$\mathbb{R}^{d_{\text{in}}}$
y_i	Target label / next token	scalar / categorical
W_0	Frozen base weight matrix	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$
ΔW_e	LoRA weight of expert e	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$
A	LoRA down-projection matrix	$\mathbb{R}^{r \times d_{\text{in}}}$
B	LoRA up-projection matrix	$\mathbb{R}^{d_{\text{out}} \times r}$
$G_{i,e}$	Routing weight for expert e at token i	scalar
V	Expert-specific neuron query vector	$\mathbb{R}^{d_{\text{out}}}$
M	Binary neuron activation mask	$\{0,1\}^{d_{\text{out}}}$
$\Delta \hat{W}_e$	Masked LoRA weight of expert e	$\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$
h_i	Original hidden output at token i	$\mathbb{R}^{d_{\text{out}}}$
$\hat{h}_i$	Compressed hidden representation at token i	$\mathbb{R}^{d_{\text{out}}}$
N	Number of experts	scalar
k	Number of activated experts	scalar
β	Sparsity regularization coefficient	scalar
γ	Diversity loss coefficient	scalar
λ	Balance loss coefficient	scalar
π	Bernoulli prior probability	scalar

benchmark, and the results are reported in Table 6. As shown in the table, although increasing the number of experts leads to improved performance, the gains are marginal, indicating that introducing more trainable parameters exacerbates knowledge redundancy and does not yield the expected performance benefits.

D.3 Analysis of Visualization

IBMoE aims to achieve fine-grained expert specialization through the SNA and NC modules. To illustrate this effect, we conduct an experiment to visualize the token embeddings processed by different experts. Specifically, we randomly select a batch of samples and process them using both sparse-routing MoE baselines and IBMoE. The outputs of each expert in a given MoE layer are collected as the visualization dataset. As shown in Figure 8, tokens processed by vanilla MoE methods are more dispersed and exhibit unclear boundaries between experts, yielding substantial overlap near the center (Figure 8(a)). In contrast, IBMoE exhibits clearer boundaries across experts, with tokens assigned to each expert forming more compact clusters (Figure 8(b)). This demonstrates that IBMoE reduces knowledge redundancy while enhancing expert spe-

Table 5: Full ablation results of IBMoE.

Methods	Act Param ↓	Datasets							Avg
		MRPC	CoLA	RTE	SQA	OBQA	CSQA	Convai	
IBMoE	7.89M	87.25	87.15	88.43	93.34	82.40	79.77	32.09	78.63
$-L_b$	7.91M	85.79	86.86	87.95	93.31	82.00	78.90	30.36	77.88
$-L_b - L_s$	7.07M	85.91	86.57	87.06	93.20	81.40	79.10	31.61	77.83
$-L_b - L_s - L_d$	13.16M	84.63	86.28	86.29	93.03	80.80	78.20	31.45	77.23

Table 6: The performance of IBMoE with different expert settings.

Expert Settings	GLUE Benchmark			Average
	MRPC	CoLA	RTE	
1 shared, 4 specific, top-2	87.25	87.15	88.43	87.61
1 shared, 8 specific, top-3	87.06	87.55	89.17	87.93

cialization. The token embeddings assigned to each expert are projected to two dimensions using t-SNE and plotted with a distinct color for each expert.

Considering that different layers process different levels of semantic information, we select multiple MoE layers to measure differences among experts. We introduce two measures: the similarity of processed information within each expert, denoted as **inner-expert similarity**, and the similarity between experts, denoted as **inter-expert similarity**. Let the tokens allocated to the i-th expert be X_{e_i}, and let the expert's knowledge embedding be defined as $\text{KE}_i = \text{mean}(E_i(X_{e_i}))$. The two similarity measures are defined as follows.

Inner-expert similarity (cohesion of tokens processed by the same expert):

$$\text{InnerSim}_i = \frac{1}{|X_{e_i}|} \sum_{x \in X_{e_i}} \text{cosine}(E_i(x), \text{KE}_i) \tag{19}$$

Inter-expert similarity (redundancy between experts):

$$\text{InterSim} = \frac{2}{N(N-1)} \sum_{1 \le i < j \le N} \text{cosine}\left(\text{KE}_i, \text{KE}_j\right) \tag{20}$$

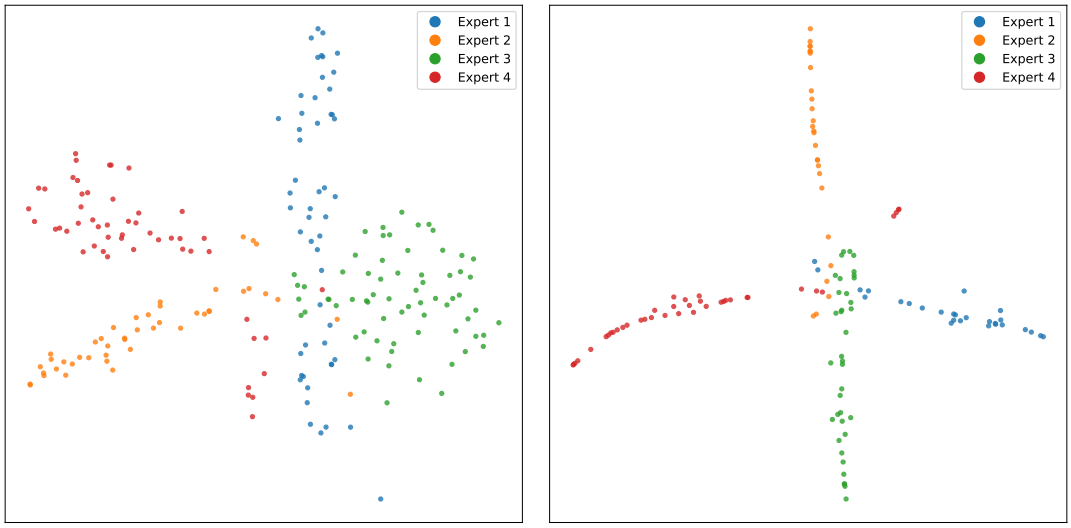
The results are presented in Table 7, where every six layers of the MoE are selected to compute the similarity. We observe that IBMoE achieves superior average performance on both measures. In particular, the inter-expert similarity of IBMoE is nearly zero, indicating that our method effectively promotes stronger expert specialization through collaborative learning. We also visualize the similarity training dynamics in Figure 9. In the whole training process, Our method almost has the higher inner-similarity and lower inter-similarity compared to the vanilla MoE method. Moreover, we find that higher similarity tends to occur in the lower layers, which primarily process general semantic knowledge, while the upper layers that capture task-specific knowledge typically exhibit lower similarity. This observation is consistent with previous findings (Gao et al., 2024).

D.4 Effect of Output Scaling.

We note that the dynamic neuron sparsity introduced during training may potentially affect stability. To address this issue, we introduce a scaling term $d_{out}/sum(M_e)$ in Equation 10, which serves a similar role as the scaling factor in LLaMA-MoE to maintain output stability. To evaluate its effectiveness, we set this term to a constant value of 1.0 and train IBMoE on all datasets from the GLUE benchmark. The results, shown in Table 8, indicate that removing the scaling term leads to a performance drop in IBMoE.

D.5 Analysis of the Training Complexity

Considering that IBMoE introduces additional complexity in computing the loss functions compared with vanilla MoE methods, we also present a brief theoretical analysis along with empirical results regarding its computational cost. Specifically, let the input dimension be N (e.g., 11,048 in LLaMA2-7B), the number of experts be K, and the rank of a LoRA expert be r (e.g., $r = 64$). The computational complexity of a vanilla LoRA expert is $O(NrN)$. In IBMoE, each expert maintains



(a) Vanilla Sparse MoE

(b) IBMoE

Figure 8: Each expert first processes the tokens allocated by the routing mechanism. The resulting token embeddings are then visualized using the t-SNE algorithm, where points of different colors represent token representations processed by different experts.

Table 7: The cosine similarity of experts across different MoE layers. For **Inner-Expert Similarity**, we compute the similarity between the representations of tokens assigned to each expert and the corresponding expert's representation centroid. For **Inter-Expert Similarity**, we compute the similarity between the representation centroids of tokens assigned to different experts across all experts.

	Layer1	**Layer7**	**Layer13**	**Layer19**	**Layer25**	**Layer31**	**Average**
			Inner-Expert Similarity ↑				
MoE	0.7729	0.9517	0.4992	0.6219	0.5628	0.5832	0.6653
IBMoE	0.9408	0.9692	0.6999	0.7917	0.3594	0.6362	**0.7329**
			Inter-Expert Similarity ↓				
MoE	0.1600	0.1852	0.1552	0.2116	0.1488	0.1609	0.1703
IBMoE	0.0615	0.0767	0.0971	0.0932	0.0868	0.0713	**0.0811**

a set of learnable parameters $V \in \mathbb{R}^{\mathbb{N}}$, and two novel loss terms, L_d and L_s, must be computed. To compute L_d, we calculate the cosine similarity for each pair of V, which involves computing the L2 norm for each V and the dot product for every pair. This results in a complexity of L_d is $O(KN) + O(K(K-1)N/2)$. For L_s, we needs to compute mean value of each V, and the complexity is $O(KN)$. Therefore, the total complexity is $O(2KN) + O(K(K-1)N/2) < O(2KN) + O(K^2N) = O(K(K+2)N)$. In practice, K is generally much smaller than N, we have $O(K(K+2)N) \ll O(NrN)$. This indicates that the additional computation introduced by the loss functions is less than that required by a LoRA adapter. Hence, our method achieves a favorable trade-off between performance improvement and computational overhead. Empirical results as shown in Table 3, it demonstrates that IBMoE achieves a comparable training efficiency with vanilla LoRA. we contribute this efficiency gains to following key factors. (1) The NC module enables collaborative learning among experts, improving parameter utilization. Unlike previous methods that insert MoE modules into both attention and MLP blocks at every layer, IBMoE achieves comparable performance by applying MoE modules only to every other MLP block, the number of additional adapter parameters is reduced, leading to improved training efficiency. (2) The SNA module implements sparse neuron activation. For instance, with a compression ratio set to 0.5, half of the neurons

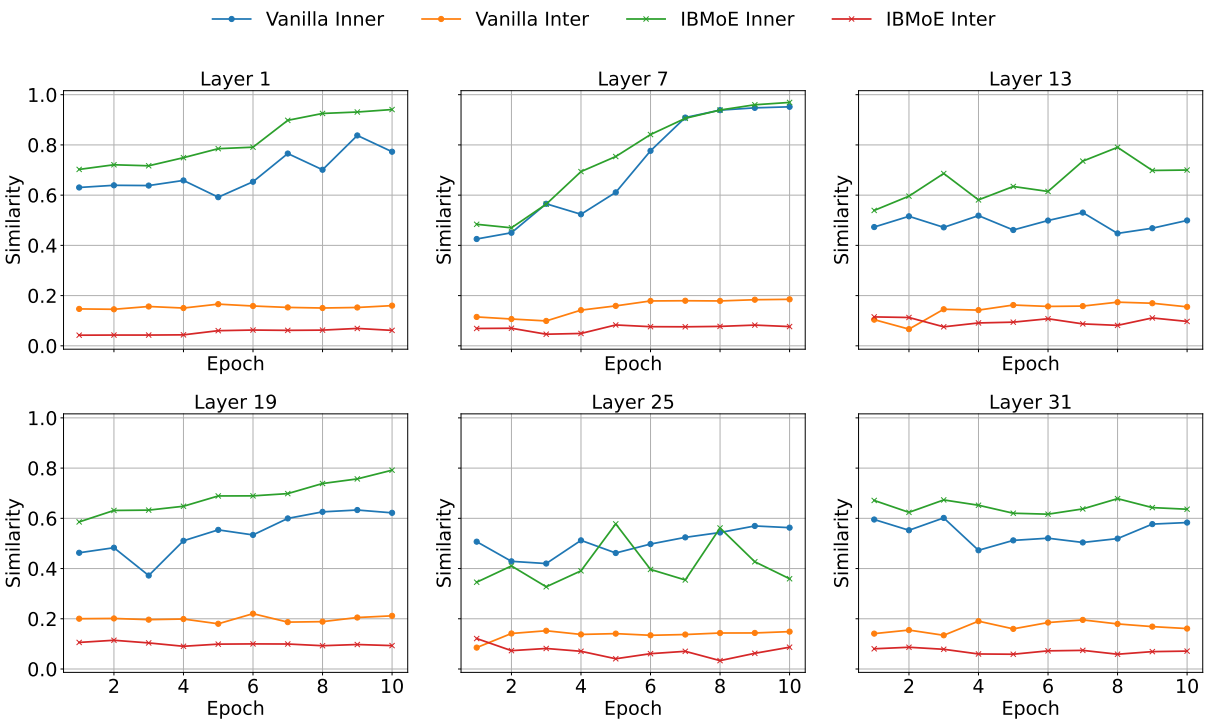


Figure 9: The training dynamics of similarity among the experts.

Table 8: The effect of output scaling.

Method	GLUE Benchmark			Average
	MRPC	CoLA	RTE	
IBMoE w/output scaling	87.25	87.15	88.43	87.61
IBMoE wo/output scaling	85.48	86.28	87.36	86.37

are deactivated during computation, significantly lowering the number of MAC operations.

D.6 Training dynamics of activated neurons

To explore how the IB-based SNA module affects neuron-level sparsity throughout the training process, we train IBMoE on the RTE dataset for 15 epochs with the compression ratio π set to 0.6. At the end of each epoch, we evaluate IBMoE on the validation set and record the number of activated neurons for each expert. As shown in Figure. 10, we use the average number of activated neurons across all experts to quantify the model's sparsity. We observe that the number of activated neurons rapidly decreases to the expected level during the early stage of training and remains stable throughout the subsequent epochs. It suggests that parameter redundancy exists at the initialization of adapters, and that each expert can quickly prune a large number of neurons under the sparsity constraint during early training, before gradually stabilizing the importance of the remaining neurons for their respective tasks.

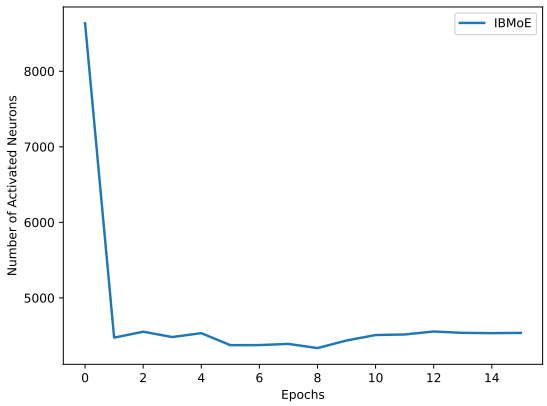


Figure 10: Training dynamics of activated neurons.

Algorithm 2 Forward Process of an IBMoE Layer.

Require: (1) A batch of input X; (2) Base module W_0; (3) LoRA weights of experts $\{\Delta W_1, \Delta W_2, \ldots, \Delta W_N\}$ with learnable queries $\{V_1, V_2, \ldots, V_N\}$; (4) Router W_r; (5) Related hyper-parameters, including temperature τ and ϵ.

$P \leftarrow \mathrm{TopK}(\mathrm{Softmax}(W_r^\top X))$
Dispatch tokens to experts $\{X_{E_1}, X_{E_2}, \ldots, X_{E_N}\}$ with corresponding routing probabilities $\{P_{E_1}, P_{E_2}, \ldots, P_{E_N}\}$ from P
$O \leftarrow W_0 X$
for $i = 1$ to N **do**
  $M_i \leftarrow \mathrm{GSSampleMask}(V_i, \tau, \epsilon)$
  $s \leftarrow d_{\text{out}}/\mathrm{sum}(M_i)$
  if mode $=$ train **then**
    $O_{E_i} \leftarrow s \cdot P_{E_i} \cdot M_i \cdot (\Delta W_i X_{E_i})$
    Aggregate expert output O_{E_i} into O
  else
    Compress ΔW_i into $\Delta \hat{W}_i$ via mask M_i
    $\hat{O}_{E_i} \leftarrow s \cdot P_{E_i} \cdot (\Delta \hat{W}_i X_{E_i})$
    Aggregate expert output $\hat{O}_{E_i}$ into O
  end if
end for

Algorithm 3 Loss optimization in IBMoE.

Require: (1) Training dataset $\{(x_t, y_t)\}_{t=1}^{T}$; (2) An IBMoE model IBMoE; (3) The number of activated experts k; (4) The number of IBMoE layers NL; (5) Hyper-parameters $\pi, \gamma, \beta, \lambda$.

$\{\hat{Y}, \{\hat{V}_j^i\}_{j=1,i=1}^{N,NL}, \{P^i\}_{i=1}^{NL}\} \leftarrow \mathrm{IBMoE}(X)$
$L_p \leftarrow \mathrm{CrossEntropy}(Y, \hat{Y})$
for $i = 1$ to NL **do**
  Compute the number of tokens assigned to each expert $\{c_1^i, c_2^i, \ldots, c_N^i\}$ from P^i
  $L_b^i \leftarrow 0,\ L_s^i \leftarrow 0$
  for $j = 1$ to N **do**
    $\bar{V}_j^i \leftarrow \mathrm{mean}(\hat{V}_j^i)$
    $L_s^i \leftarrow L_s^i + \frac{1}{N}\left[\bar{V}_j^i \log \frac{\bar{V}_j^i}{\pi} + (1 - \bar{V}_j^i) \log \frac{1-\bar{V}_j^i}{1-\pi}\right]$
    $L_b^i \leftarrow L_b^i + \frac{1}{N}\frac{N}{k}\frac{c_j^i}{T}\mathrm{mean}(P^i)$
  end for
  $L_d^i \leftarrow \frac{1}{N(N-1)} \sum_{j \neq k} \mathrm{cosine}(\hat{V}_j^i, \hat{V}_k^i)$
end for
$L_b \leftarrow \frac{1}{NL} \sum_{i=1}^{NL} L_b^i$
$L_s \leftarrow \frac{1}{NL} \sum_{i=1}^{NL} L_s^i$
$L_d \leftarrow \frac{1}{NL} \sum_{i=1}^{NL} L_d^i$
$L \leftarrow L_p + \beta L_s + \gamma L_d + \lambda L_b$

D.7 Compatibility with Computational Efficiency Strategies

Recently, several works have sought to reduce computational cost by modifying model architectures or optimizing the forward pass, such as pruning, compression, and merging. From a structural perspective, redundant blocks in dense models can be pruned either before or after training (He et al., 2024). From a computational perspective, expert-merging strategies maintain a constant inference cost as the number of activated experts increases (He et al., 2023). In IBMoE, we do not modify the computation process of the vanilla MoE framework, which allows existing acceleration strategies to be readily integrated into IBMoE.

In this section, we integrate an expert-merging strategy into IBMoE to explore its compatibility with other computational efficiency methods. Following (He et al., 2023), we merge multiple activated experts into a single expert for a single forward pass. The experiment is evaluated on 1,000 randomly selected samples, with the batch size set to 1, 8 and 16, respectively. The results in Table 9 show that the merging strategy based on sequence-level expert selection can provide computational acceleration significantly. We attribute this behavior to the fact that this strategy avoids the cost of dispatching tokens to individual experts, as in vanilla token-level MoE, and that the padded inputs naturally enable more efficient parallel computation within a single forward pass.

In summary, we provide a preliminary exploration of combining IBMoE with other acceleration methods, which demonstrates the strong compatibility of IBMoE in practical deployment scenarios.

Table 9: Inference time comparison with and without expert merging.

Methods	Batch Size	Inference Time (s)
IBMoE	1	868.94
w/merging	1	432.31
IBMoE	8	148.27
w/merging	8	81.41
IBMoE	16	96.45
w/merging	16	60.64